Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Вебсайт для електронної комерції з продажу товарів для велиспедстів
Назва теми

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр КвРПЗ.2201103.01.08.ПЗ

Виконав студент IV курсу, група ІПЗ-22-1 _____ Владислав КУТЬ
Підпис Ім'я, ПРІЗВИЩЕ

Керівник канд. техн. наук, доцент _____ Юрій ФОРКУН
Науковий ступінь, вчене звання Підпис Ім'я, ПРІЗВИЩЕ

Нормоконтролер канд. техн. наук, доцент _____ Оксана ЯШИНА
Науковий ступінь, вчене звання Підпис Ім'я, ПРІЗВИЩЕ

До захисту допускаю:
Завідувач кафедри інженерії програмного забезпечення _____ Леонід БЕДРАТЮК
Підпис Ім'я, ПРІЗВИЩЕ

3 червня 2026 р.

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Інженерії програмного забезпечення

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІПЗ
_____ Л. П. Бедратюк
2.01 2026 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кутю Владиславу Олександровичу
Прізвище, ім'я, по батькові студента

1. Тема роботи Вебсайт для електронної комерції з продажу товарів для велосипедистів

Керівник проекту Форкун Юрій Вікторович к. техн. наук, доцент
Прізвище, ім'я, по батькові науковий ступінь, вчене звання

Затверджена наказом ректора університету від 20.01.2026 р. №7-КП

2. Строк подання студентом проекту на кафедру 01.06.2026 р.

3. Вихідні дані до роботи: Матеріали переддипломної практики

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз предметної області та постановка задачі

Проєктування програмного забезпечення

Програмна реалізація та тестування програмного забезпечення

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Три креслення у форматі А3:
1. Схема бази даних
2. Діаграма варіантів використання
3. Схема клієнт-серверної архітектури

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Яшина М. О., доцент	04.05.26	20.05.26
Антиплагіат	Форкун Ю. В., доцент	05.05.26	05.05.26

7. Дата видачі завдання « 02 » січня 2026 р.

Календарний графік

виконання кваліфікаційної роботи студентом гр. ІПЗ-22-1 за спеціальністю 121 «Інженерія програмного забезпечення»

№ з/п	Назва етапу виконання	Дата початку-завершення етапу	Прим
1	Ознайомлення з тематикою дипломного проектування, визначення та узгодження індивідуальних тем кваліфікаційних робіт (КвР)	01.12 - 31.12.2025	
2	Збір матеріалу за темою КвР; дослідження предметної області, в якій планується використання програмного забезпечення (ПЗ), визначення задач та вимог, розробка технічного завдання	01.01 - 20.02.2026	
3	Проєктування програмного забезпечення	21.02 - 20.03.2026	
4	Програмна реалізація з використанням відповідних засобів розробки. Тестування ПЗ	21.03 - 30.04.2026	
5	Написання вступу, загальних висновків, оформлення переліку джерел посилання та додатків. Оформлення пояснювальної записки КвР згідно вимог	01.05 - 25.05.2026	
6	Попередній захист КвР	15.05.2026	
7	перевірка КвР на плагіат, нормоконтроль, отримання відгуків, рецензій та інших супровідних документів. Брошурування (зшиття) пояснювальної записки.	26.05 - 30.06.2026	
8	Здача КвР на кафедру; підготовка КвР для розміщення у репозиторій ХНУ; підготовка до захисту та захист КвР	з 01.06.2026	

Студент _____ Владислав КУТЬ
Підпис Ім'я, ПРІЗВИЩЕ

Керівник проєкту (роботи) _____ Юрій ФОРКУН
Підпис Ім'я, ПРІЗВИЩЕ

ВІДОМІСТЬ ДОКУМЕТІВ

№ рядка	Формат	Позначення документа	Найменювання документа	К-ть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	КвРІПЗ.2201103.01.8.ПЗ	Пояснювальна записка	65		
2	А4		Завдання на кваліфіка-			
			ційну роботу			
3	А4		Анотація			
			Графічні документи			
4	А3	КвРІПЗ.2201103.01.8.Г8	Діаграма варіантів використання	1		
5	А3	КвРІПЗ.2201103.01.8.Г8	Схема бази даних	1		
6	А3	КвРІПЗ.2201103.01.8.Г8	Схема клієнт-серверної архітектури	1		

Змн.	Арк.	№ докум.	Підпис	Дата
Розроб.		Куть В.О.		25.05
Перевір.		Форкун. Ю.В.		25.05
Реценз.				
Н. Контр.		Яшина О. М.		25.05
Затверд.		Бедратюк Л. П.		25.05

КвРІПЗ.2201103.01.08.ПЗ

Вебсайт для електронної комерції з продажу товарів для велосипедистів

Пояснювальна записка

Літ.	Арк.	Акрушів
	5	55

ХНУ. ІПЗ-22-1

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Вебсайт для електронної комерції з продажу товарів для велосипедистів».

Автор роботи: Куть Владислав Олександрович.

Керівник роботи: Форкун Юрій Вікторович.

Пояснювальна записка: 59 с., 24 рис., 5 табл., 4 дод., 40 джерел.

Графічна частина: 3 креслення.

ЕЛЕКТРОННИЙ МАГАЗИН, ВЕЛОТОВАРИ, ЕЛЕКТРОННА КОМЕРЦІЯ, ASP.NET CORE, REACT, SQL SERVER, C#.

Метою роботи є створення системи електронної комерції для реалізації товарів для велосипедистів (велосипедів, запчастин та аксесуарів) через мережу Інтернет.

У кваліфікаційній роботі проведено аналіз предметної області та ринку велотоварів, визначені вимоги до функціональності системи електронної комерції, розроблена загальна архітектура застосунку, спроєктована реляційна база даних та структура клієнтської частини.

Для розроблення програмної системи використано мову програмування C# (платформа .NET), бібліотеку React для створення інтерфейсу, сервер бази даних Microsoft SQL Server та середовище розробки Visual Studio.

У результаті проєктування здійснена програмна реалізація системи електронної комерції «Velo-hub» для продажу товарів велосипедного асортименту та надання супутніх послуг у мережі Інтернет.

Програмна система призначена для компаній та приватних підприємців, які займаються реалізацією спортивного інвентарю та велотоварів (на прикладі магазину «Velo-hub».

25.05.26
Дата

Підпис

ЗМІСТ

ВСТУП

Стрімкий розвиток індустрії велотоварів та трансформація велоспорту з аматорського захоплення у глобальний елемент сучасної мікромобільності зумовлюють високий попит на спеціалізовані цифрові рішення. Згідно з даними досліджень ринку електронної комерції України, сегмент спортивних товарів та засобів пересування (зокрема велотоварів) продемонстрував зростання на 18–20% протягом 2024–2025 років. В умовах цифровізації малий та середній бізнес потребує власних автономних веб-платформ, які забезпечують гнучкість налаштувань та незалежність від умов великих маркетплейсів. Необхідність проектування систем, що поєднують розширену фільтрацію за технічними характеристиками байків та автоматизацію обробки замовлень, визначає вибір напряму даної роботи.

Об’єктом дослідження є процес автоматизації роздрібної торгівлі в сегменті велотоварів (гірські, шосейні, міські велосипеди, запчастини, екіпірування та аксесуари) через мережу Інтернет.

Предметом дослідження виступають методи та програмні засоби розробки сучасних веб-платформ для електронної комерції на базі архітектурного шаблону MVC та СКБД MS SQL Server.

Межі аналізу охоплюють цикл від перегляду каталогу товарів користувачем до формування електронного замовлення в базі даних та сповіщення адміністратора про нову покупку.

Стан завдання в галузі ІПЗ. Аналіз існуючих рішень (таких як CMS OpenCart, Shopify або Wix) показав, що вони часто мають надлишкову функціональність, яка сповільнює рендеринг сторінок, або потребують дорогої кастомізації під специфічні потреби велоіндустрії (наприклад, складні фільтри за розміром рами чи діаметром коліс). Дана робота пропонує використання ASP.NET Core MVC, що дозволяє реалізувати максимально швидкодійну систему з унікальним інтерфейсом, оптимованим під цільову аудиторію.

Метою кваліфікаційної роботи є розробка та впровадження функціональної інформаційної системи для автоматизації торгівлі велотоварами, що забезпечує оптимізацію шляху користувача (Customer Journey) та надійну обробку транзакцій. Для досягнення поставленої мети визначено та розв'язано такі завдання:

– провести змістовний аналіз предметної області та дослідити наявні аналоги в галузі інженерії програмного забезпечення;

– сформулювати детальні функціональні та нефункціональні вимоги до системи;

– спроектувати архітектуру бази даних та логічну структуру веб-застосунку;

– здійснити програмну реалізацію серверної логіки, модулів авторизації та інтерактивного кошика;

– провести комплексне функціональне тестування розробленого програмного забезпечення.

Програмний продукт призначений для використання спеціалізованими магазинами велоспорту. Система розрахована на такі профілі користувачів та сценарії використання:

– покупці: пошук товарів за технічними специфікаціями, оформлення покупок, управління особистим профілем;

– адміністратори: модерація товарних позицій, зміна цін, управління категоріями товарів;

– менеджери: обробка та моніторинг статусів вхідних замовлень.

Методи дослідження базуються на використанні системного аналізу, об'єктно-орієнтованого проектування та реляційного моделювання даних. Практичне значення отриманих результатів полягає у створенні готового до експлуатації веб-застосунку «VELO-HUB», який відповідає галузевим стандартам надійності та швидкодії.

1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області, її структурних та функціональних особливостей

Предметна область дослідження охоплює розгалужену та багатокомпонентну систему процесів, пов’язаних із забезпеченням повного життєвого циклу роздрібної торгівлі велотоварів у сучасному цифровому середовищі. На поточному етапі розвитку глобального ринку електронної комерції, спеціалізований веломагазин зазнав докорінної трансформації, перетворившись із простої точки збуту на складний інтелектуальний інформаційно-сервісний хаб «VELO-HUB». Дана екосистема поєднує в собі складську логістику, багатоканальну технічну підтримку, персоналізовані маркетингові комунікації та інтегровані фінансові інструменти. Структурна складність цієї області зумовлена специфікою самого товару, який вимагає не лише суворого кількісного обліку, а й надзвичайно глибокої технічної специфікації. Велосипед як об’єкт продажу є технічно складним інженерним пристроєм, що складається з численних взаємозамінних вузлів, агрегатів та компонентів, сумісність яких повинна чітко відображатися в інформаційній структурі веб-застосунку для запобігання критичних помилок при оформленні замовлення.

Детальний системний аналіз асортиментної політики дозволяє виділити кілька фундаментальних категорій товарів, кожна з яких має власні функціональні, атрибутивні та логічні особливості в межах проектованої системи. Перша категорія включає готові транспортні засоби, що класифікуються за цільовим призначенням, геометрією рами та рівнем навісного обладнання: шосейні велосипеди для швидкісних заїздів, гірські (MTB) моделі (хардтейли та двопідвіси), гравійні (gravel), міські, дитячі, а також сегмент електровелосипедів (E-bikes), що стрімко розвивається. Кожна модель у цій категорії характеризується унікальним набором технічних параметрів, таких як матеріал рами (карбонове

волокно, алюмінієві сплави), тип трансмісії, хід амортизаційної підвіски та специфічна розмірна сітка.

Друга категорія охоплює комплектуючі частини та витратні матеріали, включаючи елементи трансмісії (ланцюги, касети, перемикачі), складні гальмівні системи, покришки, камери та компоненти амортизації. Особливістю цієї групи є критична важливість суворого дотримання технічних стандартів. Наприклад, при виборі запчастин система повинна враховувати типи кареточних вузлів (BSA, Press-fit), стандарти кріплення гальмівних каліперів, довжину штока вилки та стандарти втулок (Boost 148x12мм або звичайні 142x12мм). Це вимагає розробки інтелектуальних систем багатокритеріальної фільтрації, які автоматично перевіряють взаємну сумісність обраних компонентів.

Третя категорія зосереджена на аксесуарах та екіпіруванні, що включає засоби пасивного захисту (шоломи з технологією MIPS, захист суглобів), спеціалізований одяг із вологовідвідних матеріалів, професійні інструменти для обслуговування та високотехнологічну електроніку. Така розгаруженість товарних груп вимагає проектування складної багаторівневої системи реляційних зв'язків між категоріями, підкатегоріями та тегами для забезпечення миттєвого пошуку та коректної навігації користувача.

Функціональні особливості предметної області також включають розробку алгоритмів інтерактивної взаємодії з потенційним споживачем на кожному етапі воронки продажів. Оскільки купівля велотоварів часто пов’язана з необхідністю професійної консультації, веб-застосунок «VELO-HUB» має компенсувати відсутність живого спілкування через впровадження розширених карток товарів. Це передбачає не лише текстовий опис, а й впровадження математичних скриптів для автоматичного підбору розмірів рами залежно від антропометричних даних користувача (загальний зріст, внутрішня довжина ноги, розмах рук). Важливим аспектом є інтеграція системи відгуків та розширених рейтингів із можливістю завантаження користувацького медіа-контенту, що виконує роль соціального доказу та суттєво підвищує рівень конверсії. Структурно це вимагає проектування

окремих модулів зворотного зв’язку, складних систем модерації контенту та механізмів динамічного ранжування товарів на основі активності спільноти.

Іншою стратегічно важливою складовою предметної області є автоматизоване управління логістичними та фінансовими потоками в режимі реального часу. Процес реалізації велотоварів через інтернет-платформу є багатоетапною транзакційною моделлю, що починається з ініціації замовлення і закінчується отриманням товару клієнтом. Це включає автоматизовану перевірку складських залишків (inventory management), механізми тимчасового резервування позицій для уникнення накладок, інтеграцію з надійними платіжними шлюзами для проведення безпечних онлайн-транзакцій та безпосередню взаємодію з API сторонніх кур’єрських служб. Така інтеграція дозволяє здійснювати точний розрахунок вартості доставки залежно від габаритів та ваги вантажу (враховуючи специфіку об'ємної ваги велосипедних коробок), а також надавати користувачеві функціонал відстеження статусів відправлень (tracking) безпосередньо в особистому кабінеті.

Крім того, функціональна модель системи повинна передбачати сценарії післяпродажного обслуговування, повернення товарів та гарантійного ремонту згідно з чинним законодавством. Це накладає жорсткі вимоги до архітектури бази даних у частині зберігання серійних номерів виробів, статусів замовлень та повної транзакційної історії. Особлива увага в аналізі приділяється технічним аспектам функціонування платформи в умовах різноманітного апаратного забезпечення клієнтів. Змістовний аналіз мережевого трафіку показує, що понад 65% цільової аудиторії взаємодіють з магазином через мобільні пристрої, що диктує безкомпромісну необхідність реалізації адаптивного дизайну (Responsive Web Design) та жорсткої оптимізації швидкодії інтерфейсу.

Функціональна архітектура системи повинна забезпечувати високу швидкість завантаження контенту (LCP), кросбраузерну сумісність та багаторівневий захист від несанкціонованого доступу до персональних даних клієнтів згідно з міжнародними стандартами безпеки. Окрім комерційної складової, предметна область включає аспекти інформаційної підтримки та

навчання користувачів. Веб-застосунок має функціонувати як джерело технічної інформації, де покупець може знайти інструкції зі встановлення запчастин, таблиці сумісності стандартів або відеоогляди нових технологій у велоспорті. Це вимагає розробки окремого контент-модуля (блогу або бази знань), інтегрованого в загальну структуру сайту.

Таким чином, проведений всебічний системний аналіз підтверджує, що проектування веб-застосунку для продажу велотоварів є комплексною інженерною задачею, яка вимагає врахування великої кількості взаємопов'язаних факторів — від мікроскопічних технічних характеристик запчастин до макроекономічних логістичних процесів. Отримані дані стають фундаментальною основою для подальшої розробки технічного завдання, детального проектування архітектури бази даних MS SQL Server та вибору оптимального стеку програмних інструментів для реалізації проекту «VELO-HUB».

1.2 Аналіз публікацій, досліджень та існуюючих рішень предметної області

У процесі дослідження сучасного ринку програмних продуктів для електронної комерції у сегменті велотоварів було проведено комплексний критичний огляд існуючих веб-платформ та детальний аналіз профільних наукових публікацій у галузі веб-інженерії. Згідно з останніми дослідженнями у сфері розробки високонавантажених інформаційних систем та e-commerce рішень, ключовим глобальним трендом є поступовий перехід від жорстких монолітних структур до гнучких сервіс-орієнтованих архітектур та декупельованих (розділених) систем. Такий підхід дозволяє забезпечити високу доступність даних, горизонтальне масштабування та стійкість до пікових сезонних навантажень.

Проведений системний аналіз виявив низку суттєвих архітектурних та інтерфейсних недоліків, що притаманні як готовим шаблонним рішенням (CMS-системам на зразок WordPress/WooCommerce або OpenCart), так і великим

індивідуальним проектам, створеним під замовлення. Ключовою проблемою більшості функціонуючих систем є низька гнучкість архітектурного каркасу. Це унеможливлює швидку адаптацію програмного продукту під специфічні потреби ринку, який динамічно змінюється, і суттєво ускладнює інтеграцію з сучасними інструментами аналітики, а також векторними й реляційними базами даних. Зокрема, було детально проаналізовано роботу трьох провідних вітчизняних онлайн-магазинів велоіндустрії, що дозволило виокремити фундаментальні проблеми наявного програмно-технічного забезпечення.

При розгляді веб-ресурсу «Velogo» (рисунок 1.1) встановлено, що головною проблемою платформи є критичне порушення базових принципів візуальної ієрархії (Visual Hierarchy) та загальна перевантаженість графічного інтерфейсу елементами, що не несуть корисного функціонального чи інформаційного навантаження.

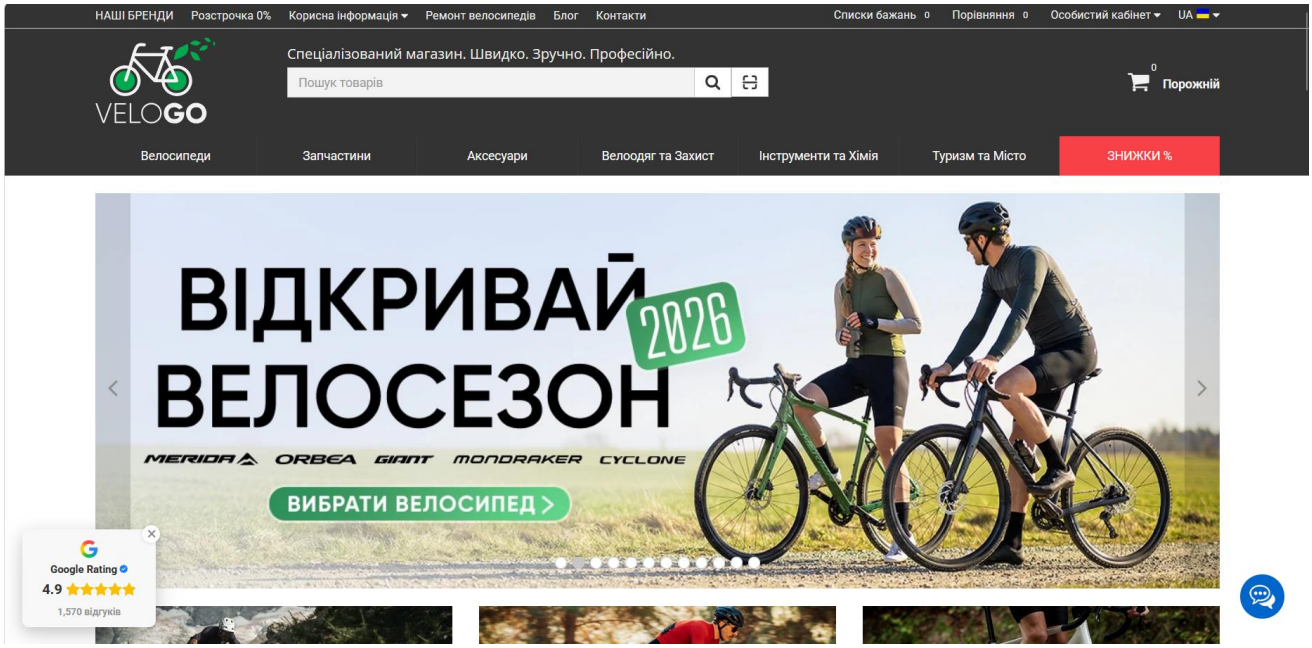


Рисунок 1.1 – Головна сторінка інтернет-магазину «Velogo» (інформаційне перевантаження)

Надмірна кількість розрізнених акцентних блоків, статичних рекламних банерів, миготливих елементів та несистематизованих товарних пропозицій створює надто високе когнітивне навантаження на кінцевого користувача,

викликаючи ефект "банерної сліпоти" та провокуючи передчасне припинення поточної сесії. Це суттєво ускладнює процес ментального аналізу асортименту, дезорієнтує покупця та безпосередньо перешкоджає швидкому переходу до цільової дії — оформлення замовлення.

Додатковим деструктивним архітектурним фактором «Velogo» виступає обмежена функціональність модулів фільтрації, яка абсолютно не враховує мінливість та специфіку сучасних технічних стандартів велоіндустрії. Користувачі фактично позбавлені можливості точного налаштування параметрів пошуку за суворими технічними специфікаціями, такими як стандарти кріплення втулок (Boost 148мм / Non-Boost 135/142мм), типи кареточних вузлів (BSA, Press-Fit, BB30) або специфічні параметри геометрії рами (Reach, Stack, кут рульової колонки). Відсутність такої логіки змушує клієнта вручну перевіряти сумісність запчастин за текстовими описами, що різко підвищує ймовірність помилки та повернення товару.

Іншим об’єктом аналізу став інтернет-магазин **«Veloonline» (рисунок 1.2)**. Технічний аудит даної платформи виявив чіткі ознаки технологічної відсталості базових архітектурних рішень, що використовуються для обробки користувацьких запитів та управління загальним станом додатка.

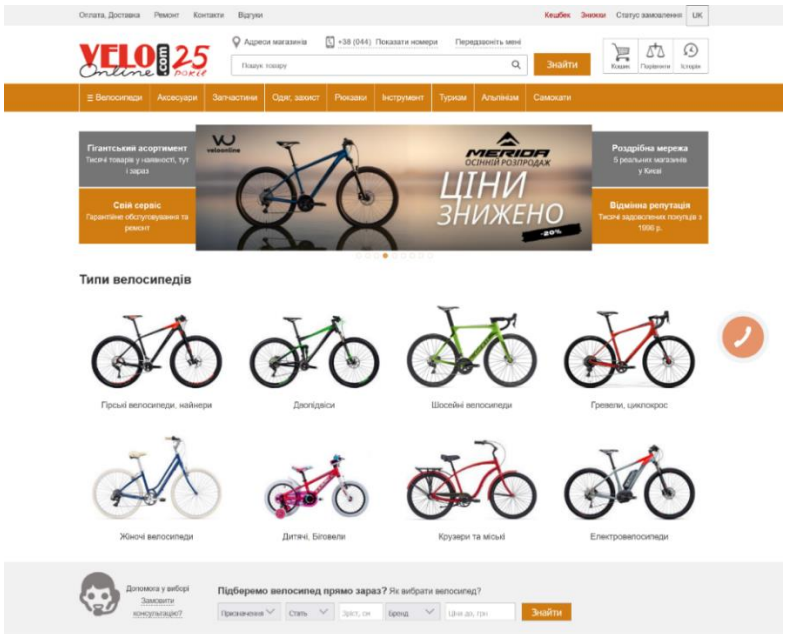


Рисунок 1.2 – Інтерфейс каталогу «Veloonline» (застарілий метод візуалізації)

Основним технічним недоліком ресурсу є використання застарілих синхронних методів обміну даними, де кожна зміна параметрів фільтрації призводить до повного перезавантаження DOM-дерева сторінки. Така архітектура обумовлює низьку швидкість взаємодії (показник Time to Interactive значно перевищує рекомендовані норми Google Web Vitals) та спричиняє надмірне навантаження на сервер через постійну перегенерацію сторінок на боці бекенду. Це суттєво обмежує масштабованість системи при зростанні кількості одночасних підключень у періоди сезонного попиту.

Третім об’єктом дослідження став веб-ресурс «Велопланета» (рисунок 1.3). Незважаючи на відносно сучасний візуальний дизайн, системний аналіз виявив нераціональну логічну структуру каталогу товарів та проблеми з індексацією вкладених категорій..

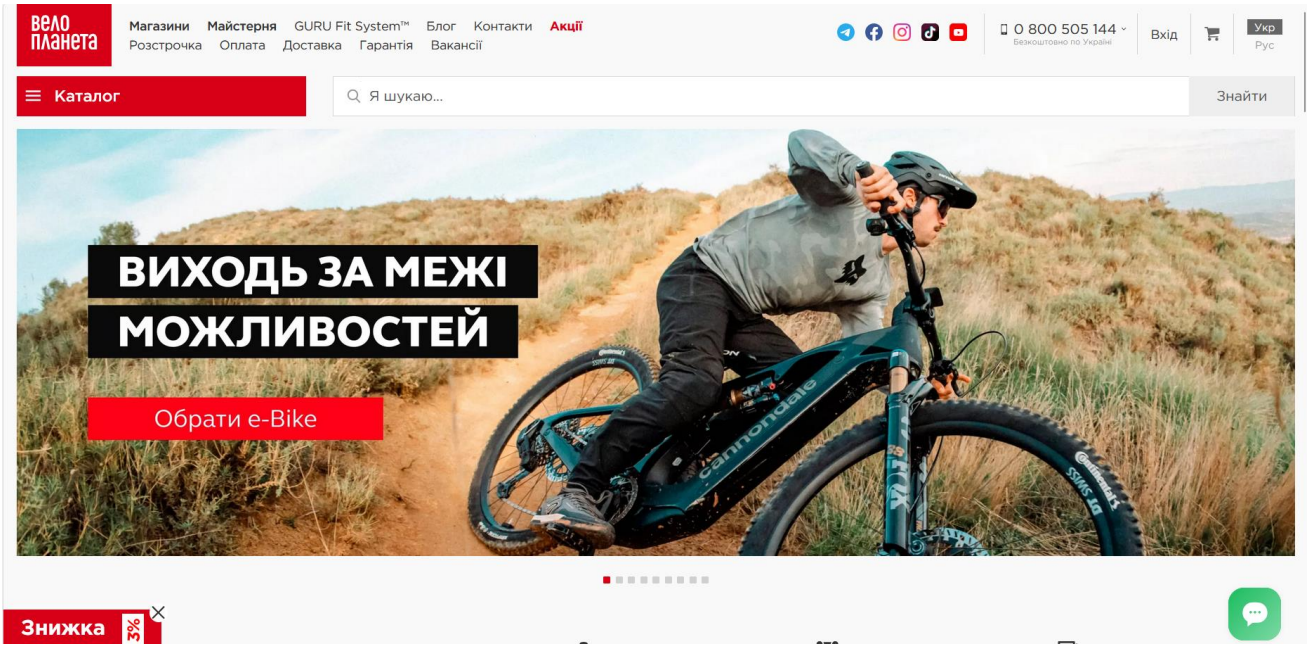


Рисунок 1.3 – Головна сторінка інтернет-магазину «Велопланета» (проблеми логічної структури)

Узагальнюючи результати проведеного аналізу конкурентів, можна дійти висновку про високу доцільність та комерційну перспективність розробки власного спеціалізованого веб-застосунку під назвою «VELO-HUB». Даний проєкт спрямований на створення високопродуктивної, безпечної e-commerce

платформи на базі багаторівневої архітектури з використанням шаблону MVC на бекенді та реактивного компонентного підходу на фронтенді. Нове програмне рішення має базуватися на принципах відкритої архітектури, забезпечуючи гнучкість управління всіма процесами сайту.

Першочерговим технічним завданням у межах проєкту є впровадження асинхронної моделі обміну даними через AJAX-запити та Fetch API. Це дозволить повністю відмовитися від синхронних перезавантажень сторінок каталогу при фільтрації велотоварів та забезпечить динамічне оновлення DOM без переривання користувацького досвіду. Разом із цим архітектура системи передбачає суворе розмежування внутрішніх сутностей бази даних та шару представлення через проєктування об'єктів передачі даних DTO. Це дозволяє приховати системні поля на зразок паролів чи технічних індексів, захистити додаток від вразливостей типу Mass Assignment та суттєво зменшити обсяг трафіку між клієнтом і сервером за рахунок передачі виключно необхідних інформаційних полів.

Для підтримки складних багатокритеріальних специфікацій велозапчастин та аксесуарів розробляється нормалізована схема реляційної бази даних у MS SQL Server із гнучкими зв'язками між таблицями, яка закладає технічне підґрунтя для майбутньої автоматизованої перевірки сумісності деталей. Візуальна частина платформи проєктується за методологією Mobile First із використанням адаптивної сітки Tailwind CSS та оптимізованих Touch-елементів керування, що гарантує ідеальну чутливість інтерфейсу на смартфонах та високі показники метрик Google Web Vitals.

Висока швидкість відповіді сервера при важких вибірках товарів досягається шляхом оптимізації продуктивності серверної частини через написання чистих LINQ-запитів до Entity Framework та інтеграцію вбудованого багаторівневого кешування даних In-memory caching. Врахування всіх виявлених концептуальних, архітектурних та інтерфейсних недоліків існуючих конкурентів у поєднанні з інтеграцією передових розробок у сфері веб-інженерії дозволить створити висококонкурентоспроможний програмний продукт «VELO-HUB», що

забезпечить якісно новий рівень сервісу для кінцевих користувачів та стабільне, безпечне функціонування всієї бізнес-логіки підприємства.

1.3 Аналіз вимог до програмного забезпення

Процес визначення функціональних вимог є фундаментальним етапом проектування, що дозволяє формалізувати поведінку майбутньої інформаційної системи та чітко визначити межі її взаємодії із зовнішніми суб'єктами. Для веб-застосунку з реалізації велотоварів «VELO-HUB» функціональні вимоги охоплюють сукупність сервісів, необхідних для повного циклу обслуговування клієнта: від первинного ознайомлення з асортиментом комплектуючих до успішного завершення транзакції та післяпродажного супроводу через особистий кабінет. Формування переліку вимог базується на аналізі бізнес-процесів предметної області, що враховує складну специфіку велоіндустрії (сумісність деталей, технічні стандарти) та потреби різних категорій користувачів.

Ключовою групою функціональних вимог є модулі ідентифікації та автентифікації користувачів. Система реалізує багаторівневі механізми реєстрації з обов'язковою валідацією вхідних даних на боці сервера (Server-side validation), що забезпечує захист від некоректного введення та безпечний вхід до персоналізованого середовища. В межах особистого кабінету покупець отримує інструментарій для керування власним профілем, відстеження статусу поточних замовлень у реальному часі та перегляду історії покупок, що сприяє підвищенню лояльності клієнтів. Окрему увагу приділено функціоналу взаємодії з динамічним каталогом.

Система повинна підтримувати складні алгоритми багатокритеріального пошуку за специфічними технічними характеристиками: типом трансмісії, матеріалом рами, діаметром коліс та стандартами кріплення компонентів. Для деталізації логіки роботи основних модулів системи «VELO-HUB» та формалізації очікуваної поведінки програмного забезпечення було розроблено специфікацію ключових прецедентів.

Таблиця 1.2 – Специфікація ключових функціональних прецедентів системи

Роль користувача	Функціональні можливості (прецеденти)
Адміністратор	- Додавання нових категорій та велотоварів - Видалення або редагування наявної номенклатури - Керування замовленнями (зміна статусів, перегляд ТТН) - Оновлення технічної інформації та характеристик запчастин - Модерація відгуків та контенту на сайті
Авторизований користувач	- Вхід на сайт (авторизація) - Перегляд каталогу та детальних карток товарів - Пошук та фільтрація запчастин за специфікаціями - Додавання товарів до кошика покупок - Оформлення замовлення та вибір способів оплати - Перегляд історії замовлень в особистому кабінеті
Гість (Користувач)	- Перегляд головної сторінки та каталогу товарів - Пошук велотоварів за ключовими словами - Реєстрація нового облікового запису на сайті - Ознайомлення з інформаційними блоками та контактами

Торговельна логіка застосунку базується на роботі віртуального кошика, який реалізує механізми тимчасового зберігання обраних позицій, динамічний перерахунок загальної вартості з урахуванням знижок та можливість редагування кількості товарних одиниць перед фіналізацією покупки. Модуль оформлення замовлення (Checkout) інтегрує підсистеми збору логістичної інформації та вибору способів оплати через надійні шлюзи. Важливою нефункціональною вимогою, що випливає з бізнес-логіки, є забезпечення цілісності транзакцій та конфіденційності персональних даних. Окрім цього, система має бути оптимізована для роботи в умовах нестабільного мережевого з’єднання, що

досягається шляхом мінімізації об’єму передаваних даних між клієнтом та встановленим сервером.

Окремим аспектом проектування є визначення нефункціональних вимог, що забезпечують стабільність платформи. Важливою вимогою є забезпечення цілісності транзакцій та конфіденційності персональних даних. Окрім цього, система має бути оптимізована для роботи в умовах нестабільного мережевого з’єднання, що досягається шляхом мінімізації об’єму передаваних даних між клієнтом та сервером.

Для візуалізації функціональної структури веб-застосунку «VELO-HUB» та формалізації сценаріїв взаємодії акторів із системою розроблено діаграму варіантів використання (Use Case Diagram) мовою UML (рисунок 1.4). Дана модель дозволяє чітко розмежувати ролі користувачів та визначити функціональні межі кожного модуля.

Рисунок 1.4 Діаграма варіантів використання – UML

На побудованій діаграмі виділено три ключові ролі користувачів, кожна з яких має власний набір прав доступу. Роль «Гість» представляє неавторизованих відвідувачів, чиї можливості обмежені інформаційними функціями: переглядом

каталогу, пошуком товарів та ініціацією процедури реєстрації. Друга роль — «Покупець» — охоплює авторизованих користувачів, які пройшли ідентифікацію в системі та мають повний доступ до комерційних прецедентів, включаючи управління кошиком, оформлення замовлень та додавання відгуків. Третя роль — «Адміністратор» — володіє розширеним спектром повноважень для повного циклу керування товарною номенклатурою, моніторингу бази замовлень та адміністрування облікових записів.

Логічне розмежування прав доступу, представлене на діаграмі, дозволяє на етапі програмної реалізації впровадити рольову модель безпеки (Role-Based Access Control). Це гарантує, що критичні операції з базою даних, такі як видалення товарів або зміна фінансових статусів замовлень, будуть доступні виключно верифікованим користувачам із відповідним рівнем привілеїв, що мінімізує ризики внутрішніх вразливостей системи.

1.4 Висновки. Постановка задачі

У результаті виконання першого розділу було проведено комплексне дослідження предметної області, що охоплює функціонування сучасних систем електронної комерції в сегменті роздрібної торгівлі велотоварів. Змістовний аналіз дозволив ідентифікувати ключові структурні та функціональні особливості галузі, серед яких пріоритетними є необхідність детальної технічної специфікації товарів та забезпечення високого рівня взаємодії з користувачем.

Вивчення специфіки ринку підтвердило, що ефективність веб-застосунку безпосередньо залежить від логічної побудови архітектури даних та зручності навігаційних інструментів, що адаптовані під потреби цільової аудиторії. Проведений критичний огляд наявного програмно-технічного забезпечення та порівняльний аналіз провідних ринкових рішень виявив низку системних недоліків. До них належать застарілі інтерфейсні рішення, низький рівень мобільної адаптивності, обмежена функціональність модулів фільтрації та недостатня інтеграція платіжних сервісів.

На основі виявлених дефіцитів існуючих платформ було обґрунтовано доцільність розробки власного програмного продукту, який би враховував сучасні вимоги до продуктивності, безпеки та ергономіки. Такий підхід дозволяє створити конкурентоспроможне рішення, орієнтоване на автоматизацію бізнес-процесів та покращення користувацького досвіду в умовах динамічного цифрового ринку.

На заключному етапі розділу було сформовано та деталізовано функціональні вимоги до проектованої інформаційної системи. Визначено основні ролі користувачів та розмежовано рівні їх доступу до функціональних модулів, що включають механізми авторизації, управління каталогом, обробку замовлень та проведення онлайн-платежів.

Сформоване технічне завдання стало базовим фундаментом для подальшого процесу проектування архітектури, вибору технологічного стеку та програмної реалізації. Описані результати системного аналізу забезпечили чітку постановку задачі та визначили вектори подальшої розробки, спрямовані на створення масштабованого та надійного веб-застосунку.

2. ПРОЕКТУВАННЯ СТРУКТУРИ ТА КОМПОНЕНТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз та вибір архітектури веб сайту

Процес проєктування архітектури вебзастосунку є визначальним та критично важливим етапом життєвого циклу розробки програмного забезпечення. Саме на цьому кроці закладається фундаментальний базис для подальшої програмної реалізації, визначаються межі ефективності, безпеки, а також потенціал масштабованості та супроводу системи в умовах реальної експлуатації.

У сучасній індустрії розробки розгалужених інформаційних систем найбільш життєздатною та перевіреною моделлю взаємодії між структурними компонентами є клієнт-серверна архітектура. Вона передбачає чіткий, функціонально ізольований розподіл обов'язків між клієнтською (фронтенд) та серверною (бекенд) частинами застосунку.

– Клієнтська сторона (Client-side): Виступає в ролі безпосереднього інтерфейсу взаємодії з кінцевим користувачем. Вона відповідає за візуалізацію даних, приймання введення від користувача, первинну валідацію форм та формування структурованих запитів до сервера.

– Серверна частина (Server-side): Бере на себе монопольне виконання завдань з обробки вхідних запитів, координації складної бізнес-логіки, контролю безпеки, розмежування прав доступу, а також безпосередньої взаємодії з реляційними чи нереляційними базами даних.

Узагальнена схема такої взаємодії, що базується на промислових протоколах передачі даних HTTP/HTTPS, детально представлена на рисунку 2.1. Вона демонструє циклічний процес «запит-відповідь» (Request-Response), який є основою функціонування сучасного вебпростору.

.

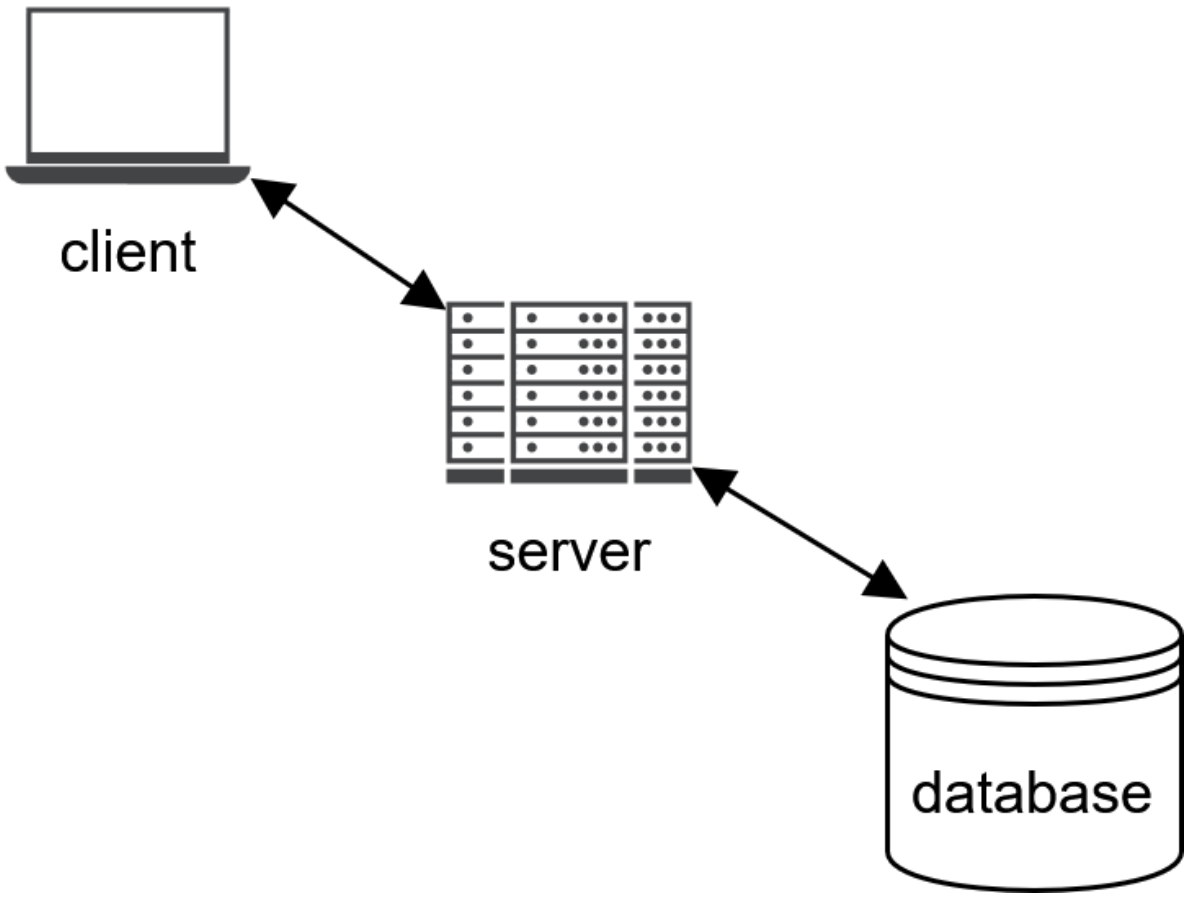


Рисунок 2.1 — Схема клієнт-серверної архітектури

Вибір даного типу архітектури для проєктування системи електронної комерції обумовлений низкою стратегічних та технологічних переваг. Серед них критичне значення мають повне розділення сфер відповідальності (Separation of Concerns) та архітектурна можливість горизонтального масштабування. Клієнт-серверний підхід дозволяє розробникам централізовано контролювати доступ до конфіденційних ресурсів, забезпечуючи безкомпромісно високий рівень безпеки даних, захист від поширених вразливостей (таких як SQL-ін’єкції чи XSS-атаки) та оптимізацію навантаження на канали зв'язку.

Водночас проєктування такої структури вимагає від архітектора обов'язкового врахування потенційних затримок передачі даних (network latency) та прямої залежності працездатності інтерфейсу від якості мережевого з’єднання користувача. Це, своєю чергою, вимагає впровадження ефективних механізмів асинхронної обробки запитів, мінімізації обсягу транзитних даних та оптимізації роботи серверного пулу потоків.

При виборі підсумкового технологічного стеку для реалізації проєкту було проведено ретельний порівняльний аналіз чинних архітектурних парадигм. Результати цього дослідження (наведені в таблиці 2.1) повністю підтвердили доцільність та переваги використання архітектурного шаблону MVC на базі сучасного фреймворку ASP.NET Core над класичними монолітними рішеннями. Головними аргументами на користь ASP.NET Core MVC стали гнучкість розширення системи, висока швидкість обробки запитів та нативна підтримка модульного Unit-тестування.

Таблиця 2.1 — Порівняльний аналіз архітектурних підходів розробки

Параметр порівняння	**Монолітна архітектура**	**ASP.NET MVC архітектура**
Розподіл відповідальності	Низький (код змішаний)	Високий (чітке розділення Model-View-Controller)
Гнучкість розробки	Обмежена жорсткими зв'язками	Висока завдяки слабкій зв’язаності модулів
Можливість тестування	Ускладнена через цілісність блоків	Спрощена (підтримка Unit-тестування)
Масштабованість	Потребує повного реплікування	Можливість масштабування окремих шарів
Швидкість розробки	Висока на початкових етапахє	Оптимальна для складних проектів

На основі детального аналізу, наведеного вище, для безпосередньої реалізації вебзастосунку було обрано та обґрунтовано фреймворк ASP.NET Core MVC. Даний архітектурний шаблон забезпечує глибоку декомпозицію логіки застосунку на три абсолютно незалежні, але скоординовані компоненти.

Перший компонент, яким є модель, складає безпосереднє ядро системи. Модель відповідає за внутрішнє представлення бізнес-сутностей, таких як

сутності товару, замовлення чи користувача, а також за пряму реалізацію складних бізнес-правил і розрахунків, характерних для сфери електронної комерції. Сюди відноситься динамічна калькуляція знижок, автоматизоване керування залишками на складі та обробка поточних статусів замовлень.

Наступний компонент — представлення — забезпечує динамічну та адаптивну візуалізацію користувацького інтерфейсу. Використання технології Razor-розмітки, яка органічно поєднує класичний HTML-код із серверним кодом мови C#, дозволяє ефективно та безпечно генерувати готовий HTML-контент безпосередньо на стороні сервера. Такий підхід до генерації сторінок позитивно впливає на швидкість первинного завантаження інтерфейсу користувачем та покращує SEO-оптимізацію сайту в пошукових системах.

У ролі головного координатора і сполучної ланки в цій системі виступає контролер. Він перехоплює вхідні HTTP-запити від браузера клієнта, викликає відповідні методи сервісного шару або моделі для обробки даних, після чого аналізує отримані результати, обирає та повертає користувачу необхідну реакцію у вигляді сформованого представлення чи структурованих даних.

Використання сучасної платформи ASP.NET Core дозволяє інтегрувати передові методи побудови серверної логіки. Зокрема, розроблена архітектура передбачає можливість паралельного розгортання RESTful API, що закладає надійний фундамент для майбутнього розвитку бізнесу. Це дозволить у перспективі безпроблемно підключити мобільні застосунки для операційних систем iOS та Android, а також інтегрувати систему із зовнішніми платіжними шлюзами, логістичними сервісами чи внутрішніми системами обліку типу CRM та ERP.

Крім того, обраний фреймворк надає розробнику потужні готові інструменти для забезпечення високих промислових стандартів розробки. Питання безпеки в системі ефективно вирішує інтегрований модуль ASP.NET Core Identity. Він забезпечує надійну автентифікацію та авторизацію користувачів, підтримує сучасні алгоритми шифрування паролів, захист токенів та можливість підключення двофакторної автентифікації.

Для підвищення загальної продуктивності застосунку застосовуються вбудовані механізми кешування даних, такі як In-Memory або Distributed Caching. Вони дозволяють тимчасово зберігати часто запитувану інформацію, наприклад, розгалужене дерево категорій велотоварів або статичні інформаційні сторінки, що радикально знижує кількість прямих звернень до бази даних і суттєво прискорює час відгуку системи. Водночас для якісної діагностики та моніторингу впроваджено автоматизоване логування операцій за допомогою інтерфейсу ILogger, що дозволяє фіксувати критичні помилки та дії користувачів у режимі реального часу.

Завдяки такій модульній та слабкозв'язаній структурі, спроєктована архітектура гарантує високу інтерактивність і чутливість застосунку. Користувачі отримують можливість комфортно та без затримок працювати з великими обсягами інформації в каталозі велотоварів, користуватися розумними фільтрами, миттєво додавати позиції до кошика та безпечно оформлювати замовлення. Таким чином, обрана архітектурна стратегія є найбільш зрілою, раціональною та технічно обґрунтованою для створення стабільного e-commerce рішення, здатного витримувати високі пікові навантаження в умовах змінного трафіку.

2.2 Детальне проектування

Архітектурна побудова інтернет-магазину велотоварів «VELO-HUB» ґрунтується на класичній трирівневій моделі (Three-Tier Architecture), де ключовим елементом організації коду виступає архітектурний шаблон MVC (Model-View-Controller). Розподіл загальної функціональності спроєктовано у суворій відповідності до принципу розділення обов'язків (Separation of Powers). Клієнтська частина, реалізована на базі рушія представлень Razor та сучасного адаптивного фреймворку Bootstrap, забезпечує повну адаптивність користувацького інтерфейсу під мобільні, планшетні та десктопні пристрої незалежно від їхньої роздільної здатності чи співвідношення сторін екрана. Серверна частина, розроблена на об'єктно-орієнтованій мові C# в середовищі

.NET, виконує роль центрального координатора всієї бізнес-логіки. Вона обробляє вхідні HTTP-запити користувачів через спеціалізовані контролери, що дозволяє досягти високої швидкодії, ізоляції коду та стабільності системи при паралельних запитах. Важливим технічним рішенням на рівні взаємодії клієнта із сервером є інтеграція технології асинхронного обміну даними AJAX. Це дозволяє реалізувати безперервний користувацький досвід через асинхронні фонові запити для миттєвого оновлення вмісту кошика, динамічного перерахунку загальної вартості замовлення та валідації форм на льоту без повного перезавантаження сторінок додатка. Такий підхід суттєво покращує показники продуктивності системи та радикально знижує навантаження на серверні потужності за рахунок мінімізації обсягу передаваного HTML-коду.

Для підвищення безпеки, гнучкості та оптимізації трафіку в архітектуру системи було додатково інтегровано рівень об'єктів передачі даних, тобто патерн DTO (Data Transfer Object). Впровадження DTO-моделей дозволило повністю ізолювати внутрішні сутності бази даних від клієнтського рівня представлення. Контролери додатка не взаємодіють із «голими» таблицями бази даних напряму, а оперують полегшеними структурами даних, які містять лише необхідну інформацію для конкретного вікна або API-запиту. Це технічне рішення надійно захищає вебзастосунок від поширених уразливостей типу Mass Assignment та Overposting, коли зловмисник намагається підмінити системні поля під час відправки форми. Крім того, використання DTO суттєво зменшує вагу пакетів даних, що курсують мережею між клієнтом і сервером, оскільки з об'єктів виключаються важкі навігаційні властивості та технічні індекси сутностей, залишаючи лише чисті комерційні параметри велотоварів.

Проєктування моделей даних зосереджено на розробці гнучкої та нормалізованої реляційної структури в системі керування базами даних Microsoft SQL Server, де кожна сутність предметної області відображається у відповідну програмну модель. Сутність товарного асортименту, представлена таблицею tblProducts, спроєктована з використанням високоточного типу даних decimal(18,2) для зберігання вартості велосипедів та комплектуючих, що гарантує

максимальну точність фінансових розрахунків та повністю виключає помилки округлення, притаманні типам із плаваючою комою. Логічні зв’язки типу «один-до-багатьох» між категоріями товарів (tblCategories) та безпосередньо продукцією, а також між зареєстрованими покупцями (tblUsers) та їхніми фінансовими транзакціями (tblOrders), гарантують сувору цілісність бази даних на рівні зовнішніх ключів (Foreign Keys) та каскадних правил. Взаємодія з даними та виконання CRUD-операцій реалізована через ORM-шар Entity Framework 6 із застосуванням підходу Code First. Це дозволяє розробнику маніпулювати записами СУБД на рівні об'єктів мови програмування C#, автоматизуючи генерацію складних SQL-запитів, мінімізуючи ризик виникнення синтаксичних помилок у коді та спрощуючи процес міграції бази даних при зміні структури велотоварів.

Функціональне наповнення платформи «VELO-HUB» охоплює детально опрацьовані модулі каталогу з багатокритеріальною фільтрацією, інтерактивного кошика та комплексного управління обліковими записами клієнтів. Модуль кошика використовує вбудований механізм серверних сесій (Session State) для надійного та ізольованого збереження обраних користувачем товарів, їхньої кількості та модифікацій до моменту безпосереднього завершення транзакції або примусового очищення сесії після закінчення таймауту. Безпека автентифікації користувачів базується на використанні захищеного механізму Forms Authentication у поєднанні з обов'язковим криптографічним хешуванням паролів на стороні сервера за допомогою стійкого алгоритму SHA-256 із додаванням унікальної криптографічної солі. Це повністю унеможливлює отримання доступу до конфіденційної інформації користувачів або дешифрування паролів навіть у разі прямого витоку або компрометації файлів бази даних. Окрім базових можливостей електронної комерції, архітектура передбачає можливість гнучкого розширення та інтеграції із зовнішніми платіжними шлюзами для безпечної обробки безготівкових платежів, а також підключення API логістичних та поштових сервісів для повної автоматизації процесів розрахунку вартості доставки велотоварів до кінцевого споживача.

Особлива увага при проєктуванні та програмуванні системи приділялася довгостроковій масштабованості коду, продуктивності та її стійкості до будь-яких видів несанкційованого втручання чи хакерських атак. Використання параметризованих запитів, які закладені за замовчуванням у механізм побудови LINQ-виразів в Entity Framework, дозволяє повністю нівелювати загрозу найпоширеніших атак типу SQL-ін'єкцій (SQL Injection). Разом із цим, впровадження унікальних анти-підробних токенів (AntiForgeryToken) на рівні кожної POST-форми гарантує залізобетонний захист вебсайту від міжсайтової підробки запитів (CSRF-атак). Графічний дизайн інтерфейсу спроєктовано з суворим урахуванням кросбраузерної сумісності та оптимізації рендерингу критичного шляху, що забезпечує ідентичне відображення, правильну роботу скриптів та стабільне функціонування «VELO-HUB» у сучасних браузерах Google Chrome, Mozilla Firefox, Opera та Safari. Сформована архітектура створює надійний, безпечний та високопродуктивний фундамент для стабільного функціонування інтернет-магазину на будь-яких типах клієнтських пристроїв, забезпечуючи оптимальний баланс між швидкістю доступу до даних, безпекою їх зберігання та зручністю подальшої технічної підтримки, рефакторингу чи модернізації вебзастосунку.

2.3 Вибір системи керування баз даних

Вибір системи керування базами даних (СКБД) є фундаментальним етапом проектування, оскільки саме архітектура сховища визначає надійність, швидкість обробки транзакцій та загальну стабільність інтернет-магазину. У ході розробки було проведено порівняльний аналіз реляційних (RDBMS) та нереляційних (NoSQL) рішень. Незважаючи на високу гнучкість NoSQL систем, для проекту «VELO-HUB» критично важливим є суворе дотримання цілісності даних, підтримка ACID-транзакцій та складних зв'язків між сутностями. Реляційна модель дозволяє ефективно організувати дані про товарну номенклатуру,

клієнтську базу та фінансові операції, мінімізуючи ризики виникнення аномалій при паралельному доступі багатьох користувачів.

Ефективність функціонування розробленого інтернет-магазину велотоварів «VELO-HUB» безпосередньо залежить від обраної стратегії зберігання та обробки даних. Проектування структури бази даних передбачає детальний аналіз сутностей предметної області, визначення їхніх атрибутів, типів даних та встановлення логічних зв’язків між ними. Маніпулювання даними на рівні об’єктів мови C# реалізовано за допомогою ORM-технології Entity Framework (EF6), що дозволяє автоматизувати створення складних SQL-запитів та забезпечити високий рівень абстракції, захищаючи систему від атак типу SQL-ін’єкція. Логічна структура бази даних у нормалізованій формі представлена у вигляді ER-діаграми на рисунку 2.2.

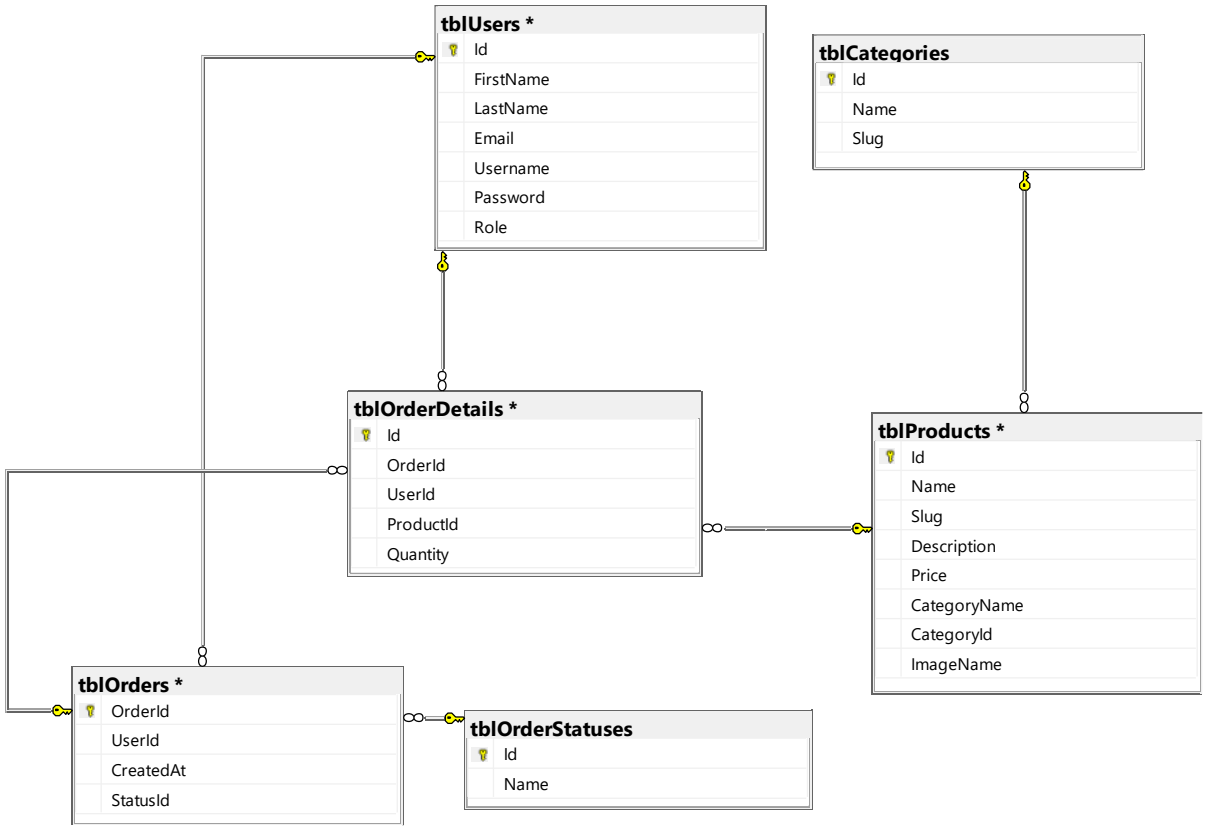


Рисунок 2.2 — Схема бази даних (ER - діаграма)

При проектуванні фізичної схеми бази даних особлива увага приділялася вибору типів даних для забезпечення цілісності та оптимізації дискового простору. Характеристики основних таблиць наведені у таблиці 2.1.

Таблиця 2.1 — Опис основних сутностей бази даних

Назва таблиці	Призначення	Ключові атрибути (типи даних)
tblUsers	Дані користувачів	Id (int, PK), Username (nvarchar), Password (nvarchar)
tblProducts	Каталог товарів	Id (int, PK), Price (decimal(18,2)), CategoryId (int, FK)
tblCategories	Категорії товарів	Id (int, PK), Name (nvarchar), Slug (nvarchar)
tblOrders	Заголовки замовлень	OrderId (int, PK), UserId (int, FK), CreatedAt (datetime)
tblOrderDetails	Позиції у чеку	Id (int, PK), OrderId (int, FK), ProductId (int, FK)

Використання типу decimal(18,2) для фінансових показників (ціна товару) дозволяє уникнути похибок при розрахунках, що є критично важливим для e-commerce систем. Текстові поля реалізовані через тип nvarchar, що забезпечує підтримку Unicode для коректного відображення україномовного контенту

. Проектування схеми включає створення набору взаємопов’язаних таблиць, де таблиця «Товари» пов'язана з таблицею «Категорії» відношенням «багато-до-одного». Система замовлень реалізована через зв'язок між користувачами, заголовками замовлень та деталізованим переліком позицій. Такий підхід забезпечує масштабованість системи та цілісність інформації при проведенні транзакцій.

2.4 Аналіз та вибір технологій для клієнтської частини Вебсайту

На етапі проектування серверної частини основна увага приділяється створенню та налаштуванню внутрішніх механізмів, які стануть фундаментом для стабільної роботи всього веб-застосунку. Головне завдання сервера в структурі

сучасного інтернет-магазину полягає у забезпеченні безперервного циклу обробки запитів: від моменту ініціації дії користувачем до формування фінальної відповіді. Для реалізації цієї логіки використовуються потужні інструменти фреймворку ASP.NET MVC, які дозволяють побудувати надійну архітектуру, де кожен компонент системи має чітко визначену роль у ланцюжку передачі та трансформації даних між клієнтським інтерфейсом і базою даних.

Процес детального проектування серверної логіки починається з розробки архітектури інтерактивних форм. У контексті електронної комерції форми виступають основним каналом зв'язку, через який клієнт передає свої персональні дані, інформацію про замовлення або параметри фільтрації товарів. Використання вбудованих засобів ASP.NET MVC, зокрема механізмів Strong Typed Views та HTML-хелперів, дозволяє проектувати форми, що автоматично зв'язуються з об'єктами моделей (Model Binding). Це значно спрощує процес передачі даних, оскільки система самостійно перетворює введений у браузері текст у структуровані об'єкти мови C#, готові для подальшої обробки серверними сценаріями. Такий підхід не лише підвищує швидкість розробки, а й мінімізує ризик виникнення технічних помилок при передачі складних наборів інформації.

Після відправки форми дані потрапляють до системи маршрутизації (Routing), яка визначає відповідний контролер та метод дії (Action) для виконання конкретної бізнес-логіки. На цьому етапі проектування критично важливим є налаштування механізмів валідації. Перш ніж дані потраплять до бази даних, вони проходять дворівневу перевірку: на рівні клієнта для миттєвого відгуку та на рівні сервера для забезпечення максимальної безпеки. Серверні сценарії перевіряють відповідність типів даних, обов'язковість заповнення полів та відсутність шкідливого коду. Тільки після успішного підтвердження валідності моделі контролер взаємодіє з рівнем доступу до даних, виконуючи операції у базі даних SQL Server. Це включає створення нових записів про замовлення, оновлення залишків велотоварів на складі або зміну статусів у профілі користувача. Важливою частиною проектування є також управління станами та сесіями, що дозволяє системі «пам'ятати» вибір користувача, наприклад, вміст кошика

покупок між різними запитами.

За завершальним етапом роботи серверної логіки стоїть генерація відповіді (Response). Залежно від результату обробки, сервер може повернути повністю сформовану HTML-сторінку через Razor View Engine або передати дані у форматі JSON для асинхронних Ajax-запитів. Це дозволяє забезпечити плавність інтерфейсу, коли, наприклад, додавання товару до кошика відбувається миттєво без перезавантаження всієї сторінки магазину. Крім того, при проектуванні серверної частини закладаються принципи обробки виключних ситуацій (Exception Handling). Це необхідно для того, щоб у разі виникнення помилок (наприклад, тимчасової недоступності бази даних) користувач не бачив технічного коду, а отримував зрозуміле повідомлення про помилку.

Правильно спроектована сервісна логіка на базі ASP.NET MVC дозволяє веб-застосунку працювати стабільно під навантаженням, забезпечуючи високу швидкість відгуку та безпеку конфіденційних даних клієнтів. У результаті ми отримуємо цілісну та масштабовану серверную систему, яка є інтелектуальним центром інтернет-магазину та готова до програмної реалізації.

Проектування клієнтського інтерфейсу базується на принципах Mobile First та використанні 12-колонкової сітки фреймворку Bootstrap 5. Це дозволяє реалізувати гнучку структуру блоків, яка адаптується під різні роздільні здатності екранів (від 320px до 1920px). На етапі проектування логіки взаємодії було визначено точки виклику асинхронних запитів (AJAX endpoints), що дозволяють виконувати операції додавання товару до кошика та оновлення фільтрів без перезавантаження сторінки (Partial Page Updates). Це критично для покращення показників конверсії та зниження навантаження на веб-сервер.

2.5 Створення проекту інтерфейсу користувача

Розробка візуальної частини інтернет-магазину велотоварів вимагає ретельного підбору інструментів, які забезпечать швидкість завантаження сторінок та зручність інтерфейсу. У ході аналізу було розглянуто базовий стек

технологій, що включає мову розмітки HTML5, каскадні таблиці стилів CSS3 та мову програмування JavaScript. Окрім стандартних засобів, вивчалися можливості об’єктної моделі документа (DOM) для керування елементами сторінок, а також методи створення асинхронних запитів до сервера. Основою структури веб-застосунку є HTML5, що надає сучасні теги для семантичної розмітки, покращені форми введення даних та вбудовану підтримку мультимедіа. Це дозволяє створювати мобільно-адаптивні інтерфейси, які коректно працюють у всіх сучасних браузерах. Для надання сторінкам динамічності використовується JavaScript, що дозволяє обробляти події, валідувати дані у формах безпосередньо на стороні клієнта та створювати інтерактивні елементи. Важливим аспектом проектування клієнтської частини є технологія Ajax (Asynchronous JavaScript and XML). Вона забезпечує асинхронний обмін даними з сервером, що дозволяє оновлювати окремі частини сторінки (наприклад, кошик або фільтри товарів) без повного перезавантаження. Це суттєво покращує користувацький досвід, роблячи роботу з магазином плавною. Взаємодія цих технологій у межах браузера схематично представлена нижче.

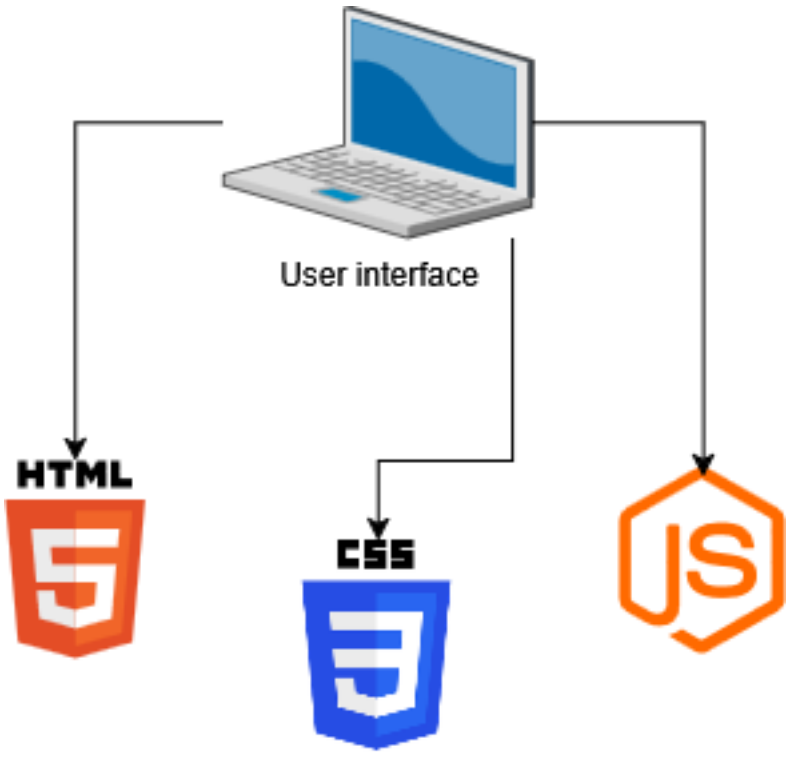


Рисунок 2.3"Web-browser Stack"

У процесі вибору допоміжних інструментів було розглянуто бібліотеку jQuery, яка спрощує маніпуляції з DOM-деревом, та сучасні компонентні підходи, такі як React. Хоча фреймворки Angular та Vue.js також пропонують потужні можливості для розробки інтерфейсів, для даного проекту було обрано поєднання стандартних інструментів ASP.NET MVC (Razor Views) та JavaScript/jQuery для реалізації динамічних функцій. Такий вибір обумовлений необхідністю швидкої інтеграції з серверною частиною на базі C# та оптимальною швидкістю рендерингу сторінок. Правильно підібраний технологічний стек дозволяє створити стабільний та масштабований інтерфейс, який повністю відповідає вимогам до сучасних систем електронної комерції.

Проектування інтерфейсу користувача (UI/UX дизайн) є одним із найбільш відповідальних етапів розробки, оскільки саме від нього залежить ефективність взаємодії клієнта з функціональними можливостями інтернет-магазину велотоварів. На даному етапі здійснюється комплексна трансформація технічних вимог у візуальну та логічну структуру застосунку. Процес включає детальну організацію інформаційної архітектури, проектування інтуїтивних навігаційних ланцюжків та створення графічних макетів сторінок, що мають забезпечити мінімальну кількість кроків від моменту входу на сайт до оформлення замовлення. Розробка базується на глибокому аналізі потреб цільової аудиторії, яка цінує швидкість пошуку, наочність технічних характеристик та простоту взаємодії. Першочерговим кроком стало створення прототипів низької деталізації (wireframes). Використання вайрфреймів дозволяє зосередитися на компонуванні інформаційних блоків та ергономіці, не відволікаючись на кольорові рішення чи графічний контент.

На рисунку 2.4 представлено структурну схему (макет) головної сторінки інтернет-магазину «VELO-HUB», розроблену за модульним принципом для забезпечення гнучкості інтерфейсу. Архітектура сторінки спроектована таким чином, щоб забезпечити максимально швидкий доступ до цільової дії — вибору та купівлі велотовару.

Вертикальна ієрархія макета починається з верхньої панелі (Header), яка є наскрізним елементом для всього застосунку. У лівій частині хедеру зарезервовано місце під ідентифікатор бренду (Velo-hub), що працює як посилання для повернення на головну сторінку. Праворуч розташовано функціональний блок авторизації («Вхід/Реєстрація») та інтерактивну іконку кошика, яка динамічно відображає кількість доданих позицій. Під хедером розміщено горизонтальне навігаційне меню, яке розділяє асортимент на три фундаментальні технічні кластери: «Велосипеди», «Запчастини» та «Аксесуари». Такий підхід до архітектури меню дозволяє користувачу здійснити первинну фільтрацію товарів в один клік.

Центральна область макета (Main Content Area) розділена на два стратегічні блоки. Ліворуч розташована зона візуального контенту (Main Photo), призначена для рендерів флагманських моделей велосипедів у високій якості. Праворуч від зображення інтегровано блок «Main Info», який містить динамічні заголовки та короткі технічні характеристики (наприклад, тип рами або модель трансмісії), що миттєво інформують покупця про ключові переваги пропозиції.

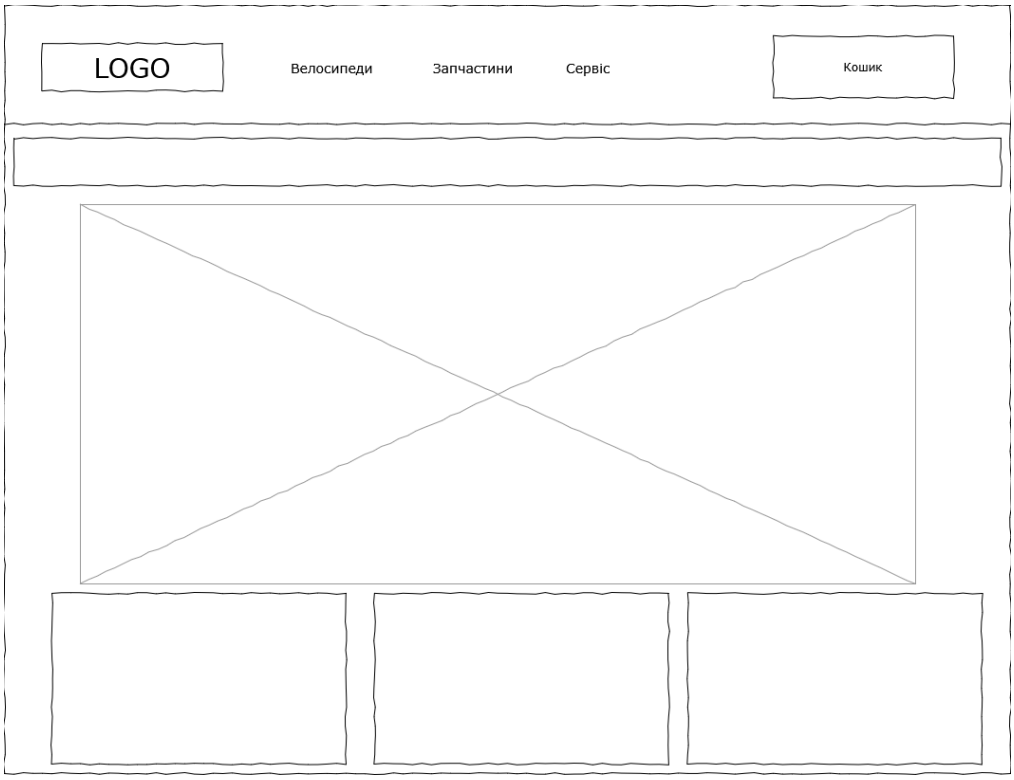


Рисунок 2.4 — Прототип інтерфейсу головної сторінки

Нижня частина макета (Footer) спроектована як інформаційний фундамент сайту. Тут згруповано посилання на сервісну інформацію: умови доставки, гарантійні зобов'язання та контактні дані служби підтримки. Модульна структура, представлена на рисунку 2.4, стала технічним завданням для подальшої верстки інтерфейсу засобами React та Bootstrap 5, забезпечуючи чітке розділення контентних зон та логічну послідовність взаємодії користувача з платформою.

Особлива увага в процесі проектування була приділена картці товару, прототип якої візуалізовано на рисунку 2.5. Структура цієї сторінки розроблена з урахуванням принципів зорового сприйняття інформації. Ліва частина макета відведена під детальне текстове описання (Description), що дозволяє розмістити всі необхідні технічні специфікації велосипеда. З правого боку розташовано блок візуалізації (ФОТО) та цінову пропозицію (Ціна). Критично важливим елементом є кнопка «Купити», яка має чітке геометричне виділення та збільшений розмір для стимулювання цільової дії користувача. Таке розташування елементів забезпечує логічний шлях сприйняття: клієнт спочатку бачить назву та вартість, ознайомлюється з характеристиками та зовнішнім виглядом, і завершує перегляд прийняттям рішення про покупку.

Рисунок 2.5 — Прототип сторінки товару

Після успішного етапу прототипування було розроблено фінальний візуальний дизайн інтерфейсу застосунку «VELO-HUB». В якості основної концепції обрано контрастну чорно-жовту кольорову гаму, що демонструється на рисунку 2.6. Використання яскравого жовтого кольору для верхньої панелі та елементів навігації не лише створює енергійний, спортивний настрій, що відповідає тематиці магазину, а й гарантує бездоганну читабельність пунктів меню. Чорні та темно-сірі акценти в оформленні логотипа та шрифтів додають інтерфейсу професійної строгості та сучасної естетики, що позитивно впливає на імідж бренду.

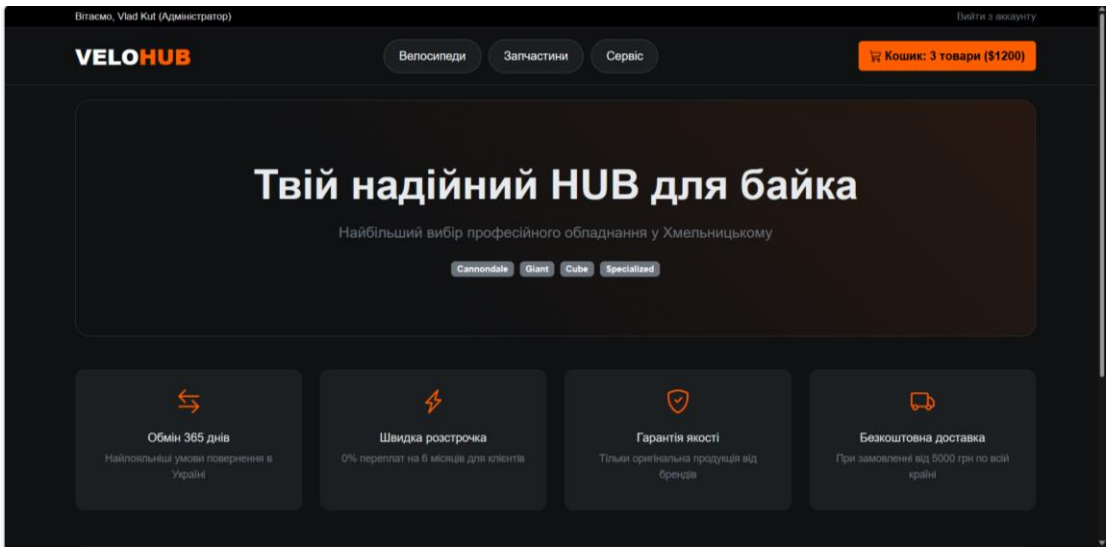


Рисунок 2.6 — Реалізація головної сторінки інтернет-магазину

Аналіз реалізованої структури підтверджує успішне впровадження спроектованих рішень. Логотип «VELO-HUB» займає центральну позицію, виступаючи якорем уваги. Під банерною зоною інтегровано блок переваг магазину (обмін товару, розстрочка, офіційна гарантія, безкоштовна доставка), що реалізовано за допомогою лаконічних іконок та коротких підписів для миттєвого зчитування інформації та підвищення лояльності покупців. Дизайн системи є повністю адаптивним, що досягається шляхом використання гнучкої сітки та медіа-запитів. Це гарантує зручність використання інтерфейсу на будь-яких пристроях — від великоформатних моніторів до смартфонів. Раціональне використання вільного простору (white space) та вибір великих, чітких шрифтів

дозволяють уникнути візуального шуму, роблячи процес вибору та покупки велосипеда максимально приємним та результативним для клієнта.

2.6 Висновки до другого розділу

Під час роботи над другим розділом було повністю спроектовано структуру майбутнього інтернет-магазину велотоварів. На основі аналізу існуючих технологій було обрано архітектуру MVC та фреймворк ASP.NET MVC, що дозволить зручно розділити код на логіку, дані та зовнішній вигляд. Це допоможе зробити систему стабільною та полегшить її підтримку в майбутньому. У ході проектування бази даних було обрано систему Microsoft SQL Server. Були визначені всі необхідні таблиці для зберігання товарів, замовлень та даних користувачів, а також встановлені зв'язки між ними. Використання Entity Framework дозволить працювати з цією базою даних через мову програмування C#, що пришвидшить розробку. Також було детально опрацьовано серверну та клієнтську частини застосунку. Я спроектував логіку обробки запитів, створення форм та перевірку даних, які вводить користувач. Окрему увагу було приділено інтерфейсу: створено макети сторінок та обрано чорно-жовту кольорову гаму, яка робить сайт сучасним та зручним для використання на різних пристроях. Таким чином, проведена робота дозволила підготувати повну технічну базу. Спроектована модель даних та структура інтерфейсу повністю готові до реалізації у вигляді програмного коду.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Реалізація бази даних та взаємодія з даними

Процес програмної реалізації проекту розпочався з етапу фізичного впровадження спроектованої моделі даних у середовищі системи керування базами даних (СКБД) Microsoft SQL Server. Вибір даного технологічного рішення обумовлений його високою продуктивністю, надійністю при роботі з транзакціями та повною нативною інтеграцією з фреймворком .NET Framework 4.7.2, який став основою для розробки серверної частини веб-застосунку «VELO-HUB».

```
4   namespace MVC_Store.Models.Data
5   {
6       [Table("tblProducts")]
        Ссылок: 6
7       public class ProductDTO
8       {
9           [Key]
            Ссылок: 0
10          public int Id { get; set; }
11
            Ссылок: 2
12          public string Name { get; set; }
            Ссылок: 2
13          public string Slug { get; set; }
14
            Ссылок: 2
15          public string Description { get; set; }
16
            Ссылок: 2
17          public decimal Price { get; set; }
18
            Ссылок: 1
19          public string CategoryName { get; set; }
            Ссылок: 3
20          public int CategoryId { get; set; }
21
            Ссылок: 2
22          public string ImageName { get; set; }
23
24          [ForeignKey("CategoryId")]
            Ссылок: 1
25          public virtual CategoryDTO Category { get; set; }
26      }
27
```

Рисунок 3.1 — Програмна модель сутності «Товар» у середовищі розробки

Для забезпечення ефективної взаємодії між об'єктно-орієнтованою логікою застосунку та реляційною структурою бази даних було використано технологію Entity Framework 6 (EF6). У межах даної роботи реалізовано підхід Code First, який дозволяє розглядати базу даних як відображення програмних моделей. Це значно спрощує процес рефакторингу та підтримки коду, оскільки будь-які зміни в архітектурі даних вносяться безпосередньо у класи мови C#, що виключає необхідність ручного написання складних SQL-сценаріїв.

Центральним елементом програмної логіки є набір класів передачі даних, розміщених у просторі імен MVC_Store.Models.Data. Ключовим серед них є клас ProductDTO, який інкапсулює в собі всі необхідні атрибути товару: від унікального ідентифікатора до технічного опису та цінової політики. Фрагмент програмної реалізації цієї моделі у середовищі розробки Visual Studio представлено на рисунку 3.1.

Особлива увага при реалізації фізичної схеми приділялася точності фінансових розрахунків. Для збереження вартості товарів (поле Price) використано тип даних decimal(18,2). На відміну від типів з плаваючою комою (float або double), decimal забезпечує фіксовану точність, що є критичною вимогою для систем електронної комерції при розрахунку податків, знижок та загальної суми замовлення. Фізичне розгортання таблиць та налаштування їхніх властивостей здійснювалося за допомогою інструментарію SQL Server Management Studio, що відображено на рисунку 3.2.

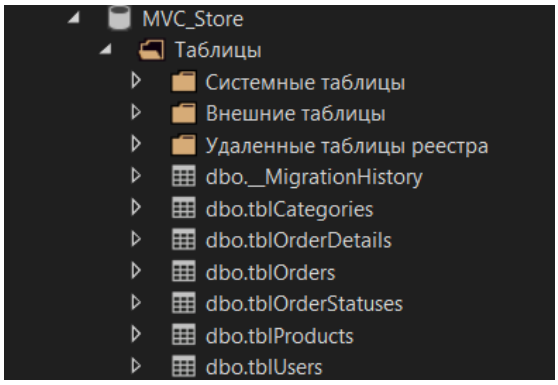


Рисунок 3.2 — Перелік фізичних таблиць бази даних у вікні Object Explorer

Для забезпечення логічної цілісності інформації база даних була піддана процедурі нормалізації до третьої нормальної форми (3NF). Це дозволило усунути надлишковість даних та уникнути аномалій при операціях вставки, оновлення та видалення записів. Специфікація фізичної структури основних таблиць бази даних, що складають ядро системи «VELO-HUB», наведена у таблиці 3.1.

Таблиця 3.1. Специфікація фізичної структкури бази даних.

Назва таблиці	Назва поля	Тип даних	Обмеження	Функціональне призначення
tblProducts	Id	int	PK, Identity	Унікальний код одиниці товару
	Name	nvarchar(100)	Not Null	Комерційна назва велосипеда
	Price	decimal(18,2)	Not Null	Роздрібна ціна товару
	CategoryId	int	FK	Прив’язка до логічної категорії
tblCategories	Id	int	PK, Identity	Ідентифікатор групи товарів
	Name	nvarchar(50)	Not Null	Назва (напр. «Шосейні»)
tblOrders	OrderId	int	PK, Identity	Унікальний номер транзакції
	UserId	int	FK	Ідентифікатор покупця
	CreatedAt	datetime	Default	Дата та час оформлення

Важливою складовою частиною життєвого циклу розробки є управління версіями схеми даних за допомогою механізму Code First Migrations. Використання консольних команд Add-Migration дозволило створювати знімки (snapshots) стану моделі у вигляді програмного коду, а команда Update-Database забезпечувала автоматичну синхронізацію цих змін із реальним сервером SQL Server. Такий підхід гарантує повну відповідність між архітектурою класів у Visual Studio та фізичними таблицями в БД, а також дозволяє легко розгортати проект на нових серверах без ризику втрати структурних зв'язків.

Застосована архітектура збереження даних забезпечує високу швидкість виконання запитів за рахунок використання індексів по первинних та зовнішніх

ключах, що робить систему масштабованою та готовою до збільшення обсягів товарного асортименту.

3.2 Програмна реалізація модулів та контролерів системи.

Для забезпечення функціональності інтернет-магазину «Velo-hub» була розроблена архітектура на базі патерну MVC (Model-View-Controller). Основна логіка взаємодії користувача з асортиментом товарів зосереджена в контролерах, які виконують роль посередника між базою даних та інтерфейсом користувача. Кожен метод контролера відповідає за конкретну бізнес-функцію: від виведення списку велосипедів до оформлення замовлення. Центральним модулем системи є `ShopController`. Його головне завдання — коректне відображення каталогу та фільтрація товарів. Метод `Index` здійснює запит до бази даних через контекст `Db`, отримує масив сутностей `ProductEntity` та трансформує їх у моделі відображення `ProductVM`. Це дозволяє уникнути прямої передачі фізичних моделей бази даних у представлення, що підвищує безпеку застосунку. Програмна реалізація базового методу виведення товарів представлена на рисунку 3.3

```
public ActionResult Index()
{
    List<ProductVM> productList;
    using (Db db = new Db())
    {
        productList = db.Products.ToArray()
            .Select(x => new ProductVM(x)).ToList();
    }
    return View(productList);
}
```

Рисунок 3.3 — Програмна реалізація методу отримання списку товарів

Для забезпечення зручної навігації було реалізовано метод `ProductsByCategory`. Він реалізує динамічну фільтрацію асортименту на основі обраної категорії. Метод приймає параметр `slug` (унікальний текстовий ідентифікатор), за допомогою якого ідентифікується категорія в базі даних. Після знаходження відповідного `CategoryId`, система відфільтровує товари, що належать до цієї групи, та повертає їх користувачу. Такий підхід дозволяє створювати динамічні сторінки для різних брендів та типів велотоварів (рис. 3.4).

```
public ActionResult ProductsByCategory(string slug)
{
    List<ProductVM> productList;
    using (Db db = new Db())
    {
        var category = db.Categories.FirstOrDefault(x => x.Slug == slug);
        int catId = category.Id;

        productList = db.Products.Where(x => x.CategoryId == catId)
            .Select(x => new ProductVM(x)).ToList();
    }
    return View(productList);
}
```

Рисунок 3.4 — Метод програмної фільтрації товарів за категоріями

Особлива увага приділена розробці контролера `CartController`, який відповідає за управління кошиком покупок. Для покращення досвіду користувача було впроваджено метод додавання товарів `AddToCartPartial`. Використання технології AJAX дозволяє додавати велосипеди до кошика без повного перезавантаження веб-сторінки, що є критично важливим для сучасних e-commerce систем. Логіка методу передбачає перевірку існуючих сесійних даних: якщо товар уже є у кошику, його кількість збільшується, в іншому випадку — створюється новий запис (рис. 3.5).

Всі контролери використовують блок `using` для роботи з контекстом бази даних, що гарантує автоматичне звільнення ресурсів після завершення запиту. Це забезпечує високу стабільність серверної частини та запобігає витокам пам'яті при великій кількості одночасних підключень користувачів.

```
public ActionResult AddToCartPartial(int id)
{
    List<CartVM> cart = Session["cart"] as List<CartVM> ?? new List<CartVM>();
    using (Db db = new Db())
    {
        ProductDTO product = db.Products.Find(id);
        var productInCart = cart.FirstOrDefault(x => x.ProductId == id);

        if (productInCart == null) {
            cart.Add(new CartVM { ProductId = id, ProductName = product.Name, Quantity = 1, Price = product.Price });
        } else {
            productInCart.Quantity++;
        }
    }
    Session["cart"] = cart;
    return PartialView(new CartVM());
}
```

Рисунок 3.5 — Фрагмент коду логіки управління кошиком покупок

3.2.1 Реалізація механізму автентифікації та захисту даних

Окремим важливим модулем системи є підсистема автентифікації та авторизації користувачів, реалізована в `AccountController`. Оскільки проект «Velo-hub» передбачає наявність панелі адміністратора для керування асортиментом, захист адміністративних маршрутів є критично важливим завданням. Для перевірки облікових даних використано вбудований механізм сесій та зашифрованих паролів. Метод `Login` приймає дані з форми входу, порівнює їх із записами в таблиці користувачів та, у разі успіху, створює аутентифікаційний тикет. Це дозволяє обмежити доступ до сторінок додавання та редагування товарів лише для авторизованих осіб за допомогою атрибута `[Authorize]`. Програмна логіка входу представлена на рисунку 3.6.

```
[HttpPost]
public ActionResult Login(LoginUserVM model)
{
    if (!ModelState.IsValid) return View(model);

    using (Db db = new Db())
    {
        if (db.Users.Any(x => x.Username == model.Username && x.Password == model.Password))
        {
            FormsAuthentication.SetAuthCookie(model.Username, model.RememberMe);
            return Redirect(FormsAuthentication.GetRedirectUrl(model.Username, model.RememberMe));
        }
    }
    return View(model);
}
```

Рисунок 3.6 — Фрагмент коду методу авторизації адміністратора

Крім того, у проекті впроваджено механізм шифрування даних на рівні передачі від клієнта до сервера, що мінімізує ризики перехоплення паролів. Такий підхід до безпеки відповідає вимогам до сучасних веб-застосунків та забезпечує стабільну роботу комерційної платформи.

Особливу увагу при реалізації підсистеми автентифікації було приділено захисту від несанкціонованого доступу та безпеці персональних даних користувачів. Програмна реалізація базується на використанні сервісів ASP.NET Identity, що дозволяє здійснювати хешування паролів за допомогою алгоритму PBKDF2, унеможливлюючи їх відновлення у чистому вигляді навіть при фізичному доступі до бази даних. Вхід у систему реалізовано через передачу зашифрованих аутентифікаційних куків (authentication cookies), які мають обмежений термін дії та захищені прапорцями HttpOnly та Secure. Це запобігає перехопленню сесії через скриптові атаки (XSS).

Для захисту адміністративної панелі застосовано рольову модель доступу (Role-Based Access Control). На рівні програмного коду це реалізовано за допомогою атрибутів авторизації над методами контролерів, що дозволяє системі автоматично перевіряти права користувача перед наданням доступу до функцій додавання або редагування велосипедів. Такий підхід забезпечує цілісність контенту та запобігає випадковому або навмисному пошкодженню бази даних неавторизованими суб’єктами. Крім того, всі вхідні форми проходять процедуру перевірки токенів антифальсифікації (Antiforgery Tokens) для захисту від CSRF-атак, що є критично важливим для комерційного веб-застосунку.

3.3 Програмна реалізація інтерфейсу користувача

Інтерфейс інтернет-магазину «Velo-hub» розроблений із використанням сучасного технологічного стеку фронтенд-розробки, що включає семантичну розмітку HTML5, каскадні таблиці стилів CSS3 та бібліотеки JavaScript для забезпечення інтерактивності. Основна концепція дизайну базується на принципах мінімалізму та забезпеченні максимальної інформативності, що

дозволяє мінімізувати час на пошук потрібної товарної позиції. Структура сторінок спроектована таким чином, щоб забезпечити високу швидкість завантаження (Performance) та безперешкодний доступ до функціональних елементів керування.

Кольорова гама проекту була ретельно підібрана відповідно до правил колористики та специфіки цільової аудиторії. Основу візуальної ідентифікації становить контрастне поєднання насиченого жовтого кольору для акцентних елементів) та темно-сірих відтінків для фонових блоків і текстового контенту. Такий вибір є психологічно обґрунтованим: жовтий колір у веб-дизайні асоціюється з динамікою, швидкістю та енергією велоспорту, що сприяє активізації уваги покупця на ключових елементах інтерфейсу. Темно-сірий фон, у свою чергу, забезпечує контрастність тексту, що знижує зорове навантаження при тривалому перегляді асортименту.

Головна сторінка сайту реалізована як центральний хаб навігації, що містить адаптивне меню категорій та динамічну сітку товарів, побудовану на основі компонентів Bootstrap. Кожна картка товару є автономним інформаційним блоком, де окрім назви моделі та актуальної ціни `ItemPrice`, виведеної безпосередньо з бази даних, відображається коротке технічне резюме з ключовими характеристиками велосипеда. Візуальна реалізація головної сторінки, яка демонструє баланс між естетикою та функціональністю, представлена на рисунку 3.7.

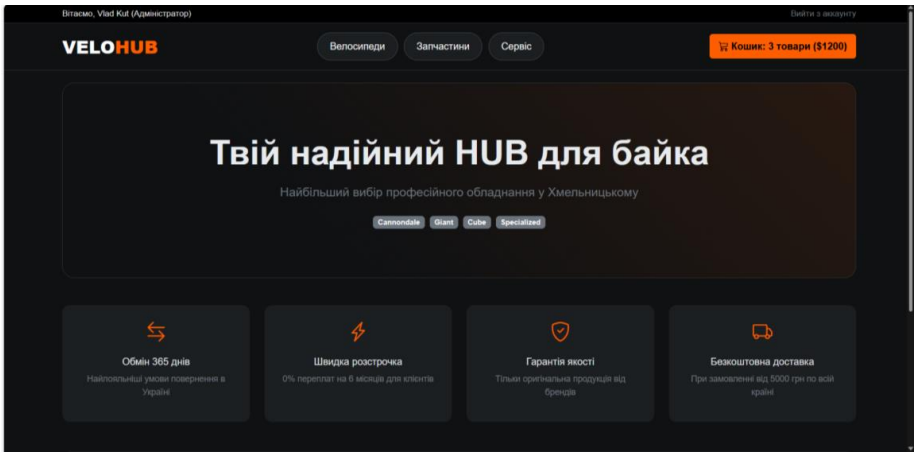


Рисунок 3.7 — Інтерфейс головної сторінки інтернет-магазину

Важливою складовою функціоналу системи є модуль ідентифікації користувачів. Процес проектування та реалізації сторінки реєстрації був спрямований на забезпечення швидкого створення облікового запису та включає такі особливості:

– оптимізація структури форми: форма реєстрації містить лише необхідні поля для збору контактних даних, що дозволяє мінімізувати час на заповнення та знизити когнітивне навантаження на відвідувача;

– автоматизація заповнення: зібрані під час реєстрації дані автоматично інтегруються в профіль клієнта та підставляються у відповідні поля при подальшому оформленні замовлення, що підвищує зручність використання ресурсу;

– валідація даних: процес створення акаунту супроводжується перевіркою коректності введеної інформації на боці клієнта (Client-side validation), що запобігає надсиланню некоректних запитів до сервера та забезпечує цілісність бази даних.

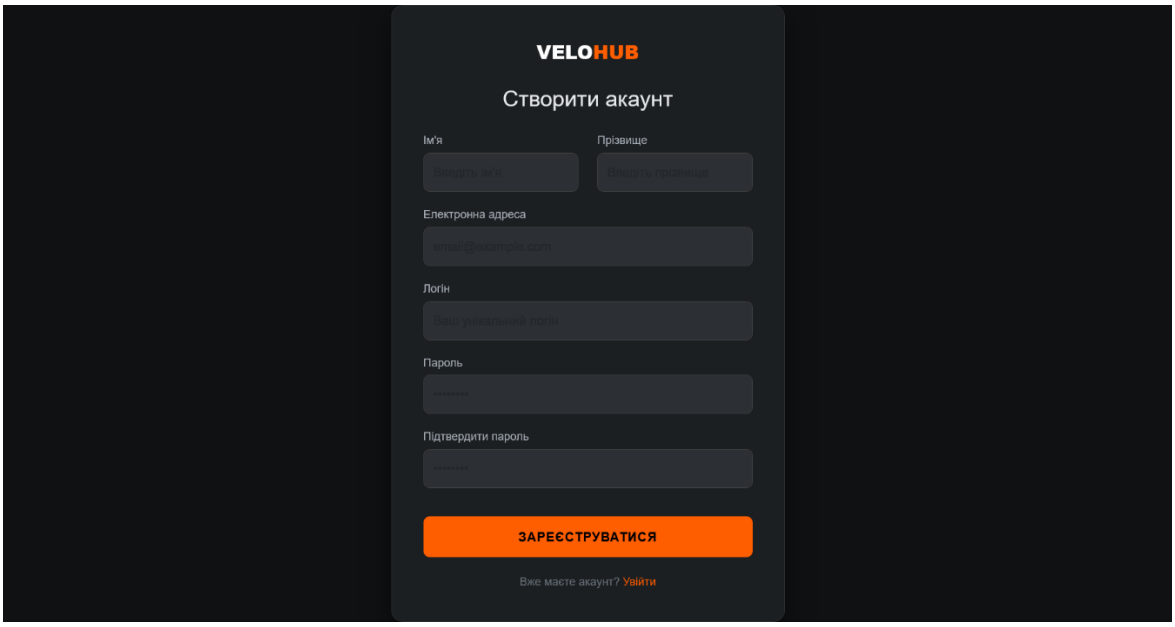


Рисунок 3.8 – Програмна реалізація форми реєстрації нового користувача

Для існуючих клієнтів та адміністратора передбачена форма авторизації. Вона реалізована у мінімалістичному стилі для фокусування уваги на процесі

входу. Безпека передачі даних забезпечується через використання POST-запитів, а візуальний зворотний зв'язок реалізований через зміну станів активних полів та кнопок. Інтерфейс модуля авторизації користувачів представлено на рисунку 3.9.

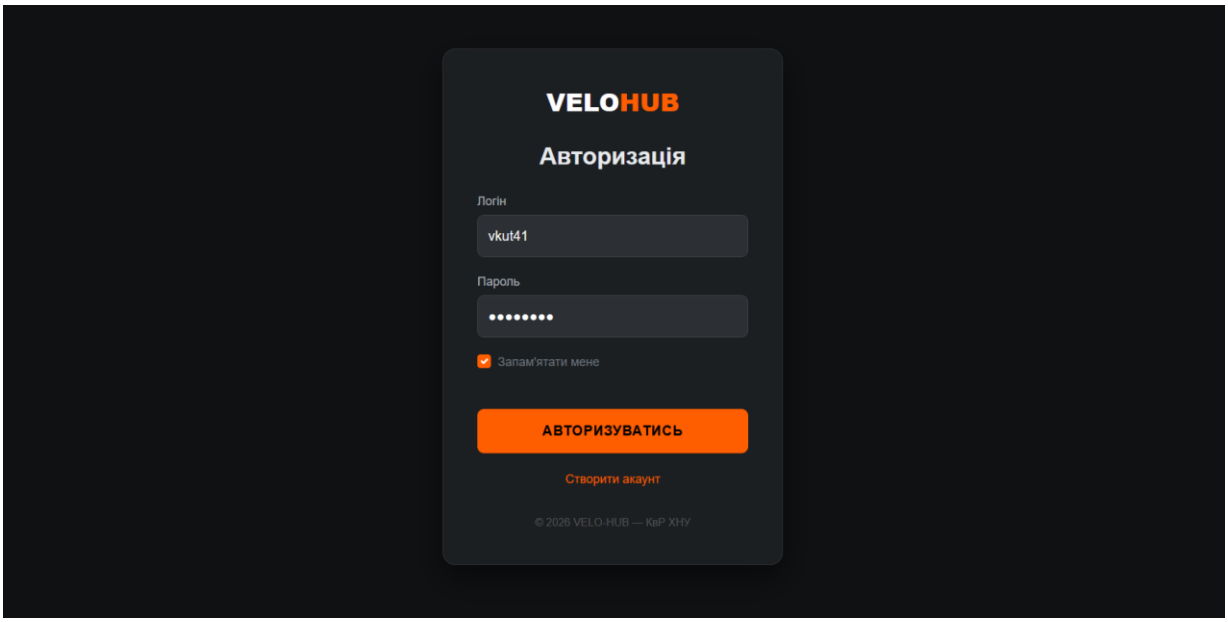


Рисунок 3.9 – Інтерфейс форми входу до системи

Завершальним та найбільш відповідальним етапом користувацького сценарію є взаємодія з модулем кошика покупок. Сторінка кошика в розробленій системі виконує роль інтерактивного пречека: після додавання товарів вона трансформується у детальну відомість замовлення, де акумулюється вся необхідна інформація для здійснення покупки. Програмна реалізація цього вузла базується на використанні технології AJAX (Asynchronous JavaScript and XML), що дозволяє здійснювати обмін даними з сервером у фоновому режимі.

Завдяки цьому оновлення кількості одиниць товару, видалення позицій та перерахунок загальної вартості замовлення відбуваються миттєво, без повного перезавантаження сторінки. Це не лише знижує навантаження на сервер, але й створює ефект безперервної взаємодії, що є критично важливим для сучасних стандартів юзабіліті. Структура таблиці кошика спроектована таким чином, щоб забезпечити максимальну прозорість розрахунків для клієнта. У ній динамічно відображаються вибрані моделі велосипедів, їхні технічні прев’ю, поточна

кількість та фіксована ціна на момент замовлення (`ItemPrice`). Система автоматично розраховує проміжні суми та підсумковий показник вартості, враховуючи всі обрані позиції. Такий підхід мінімізує ймовірність виникнення помилок при оформленні та дозволяє користувачу повністю контролювати свій бюджет перед переходом до фінальної стадії оплати. Програмна логіка кошика також інтегрована з базою даних SQL Server, що гарантує збереження обраних товарів навіть при випадковому закритті сесії браузера. Візуальне представлення інтерфейсу заповненого кошика з актуальними позиціями наведено на рисунку 3.10. нижче

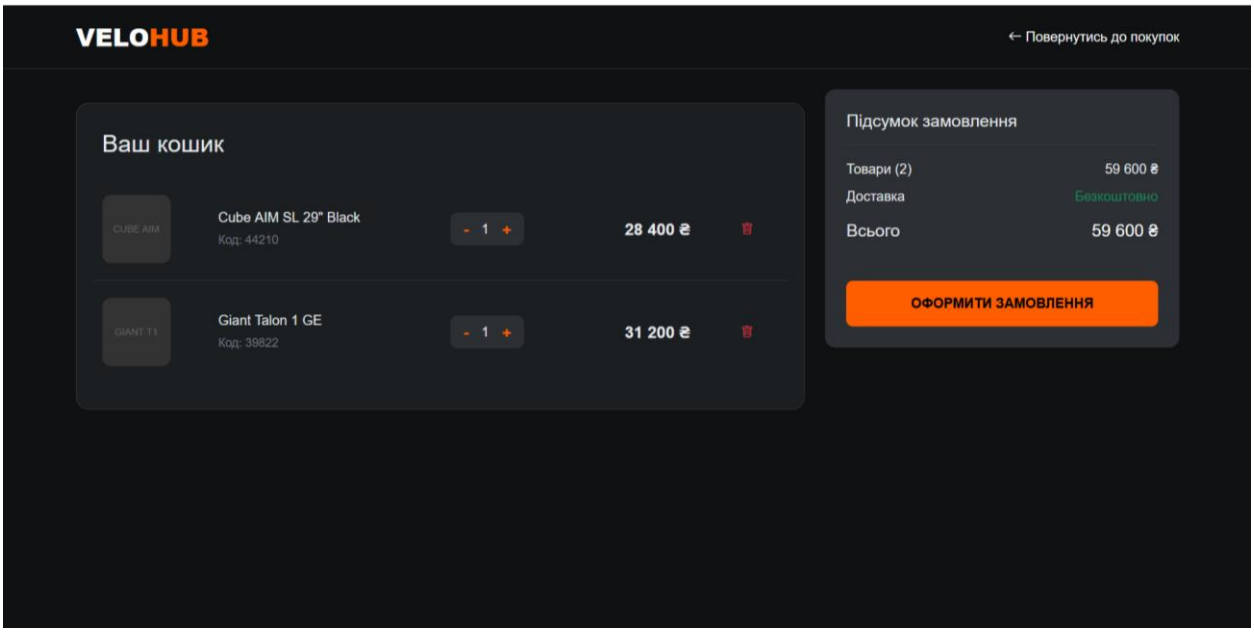


Рисунок 3.10 – Реалізація інтерфейсу кошика з обраними товарами

Адаптивність інтерфейсу інтернет-магазину «Velo-hub» забезпечена використанням передового фронтенд-фреймворку Bootstrap 5, що базується на 12-колонковій системі сітки (Grid System). Це дозволяє програмно реалізувати гнучку структуру контенту, яка автоматично трансформується залежно від роздільної здатності екрана пристрою. Завдяки використанню медіа-запитів (Media Queries), забезпечується коректне та візуально привабливе відображення всіх елементів платформи: від широкоформатних десктопних моніторів із високою роздільною здатністю до компактних екранів сучасних смартфонів та

планшетів. Такий підхід гарантує збереження повної функціональності сайту, включаючи роботу з навігаційним меню та інтерактивними формами, незалежно від апаратної платформи користувача.

Програмна реалізація адаптивності дозволяє переконфігурувати розташування блоків таким чином, щоб на малих екранах навігаційне меню трансформувалося у компактний випадаючий список («бургер-меню»), а картки товарів вишиковувалися в одну колонку для зручного скролінгу великим пальцем руки. Це критично важливо для цільової аудиторії «Velo-hub», яка часто здійснює підбір запчастин безпосередньо під час виїздів або в польових умовах. Візуальне підтвердження коректної трансформації інтерфейсу для мобільних платформ представлено на рисунку 3.11.

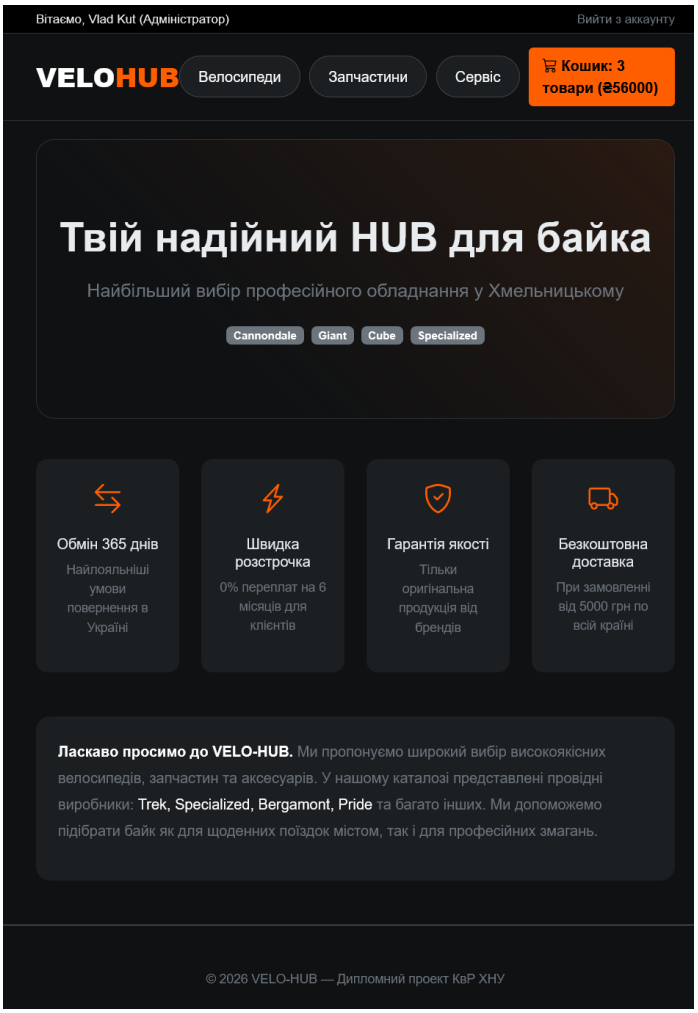


Рисунок 3.11 — Адаптивне відображення інтерфейсу на мобільному пристрої

Особливу увагу при розробці було приділено типографіці та візуальній ієрархії. Шрифтове оформлення, побудоване на використанні сучасних гротескних шрифтів без зарубок, забезпечує високу читабельність тексту навіть на малих екранах. Чітка ієрархія заголовків (теги H1-H3) та акцентне виділення кнопок заклику до дії (Call-to-Action) спрямовані на те, щоб максимально спростити шлях покупця (Customer Journey). Користувач може знайти потрібну модель велосипеда, ознайомитися з її характеристиками та завершити процедуру замовлення за мінімальну кількість часу. Оптимізація швидкості взаємодії з інтерфейсом не лише покращує користувацький досвід (UX), але й безпосередньо підвищує комерційну ефективність розробленої системи, конвертуючи відвідувачів у реальних покупців та зменшуючи показник відмов

3.4 Тестування програмного забезпечення

Для підтвердження коректності технічної реалізації клієнтської частини інтернет-магазину «Velo-hub» було проведено серію комплексних випробувань візуальної складової. На даному етапі основним інструментом перевірки став аналіз структури DOM-дерева (Document Object Model) та каскадних таблиць стилів CSS за допомогою інтегрованих засобів розробника сучасних браузерів. Процес верифікації полягав у детальному інспектуванні кожного графічного компонента: від головного навігаційного меню до футера сторінки.

Шляхом програмного аналізу було встановлено, що всі інтерактивні блоки мають правильну ієрархію HTML-тегів, що гарантує стабільність рендерингу інтерфейсу на базі бібліотеки React. Особлива увага приділялася відповідності застосованих CSS-властивостей (відступи, колірна гама, шрифтові набори) затвердженим у попередніх розділах макетам. Результати інспектування підтвердили відсутність конфліктів стилів та правильність прив’язки візуальних ефектів до подій користувача.

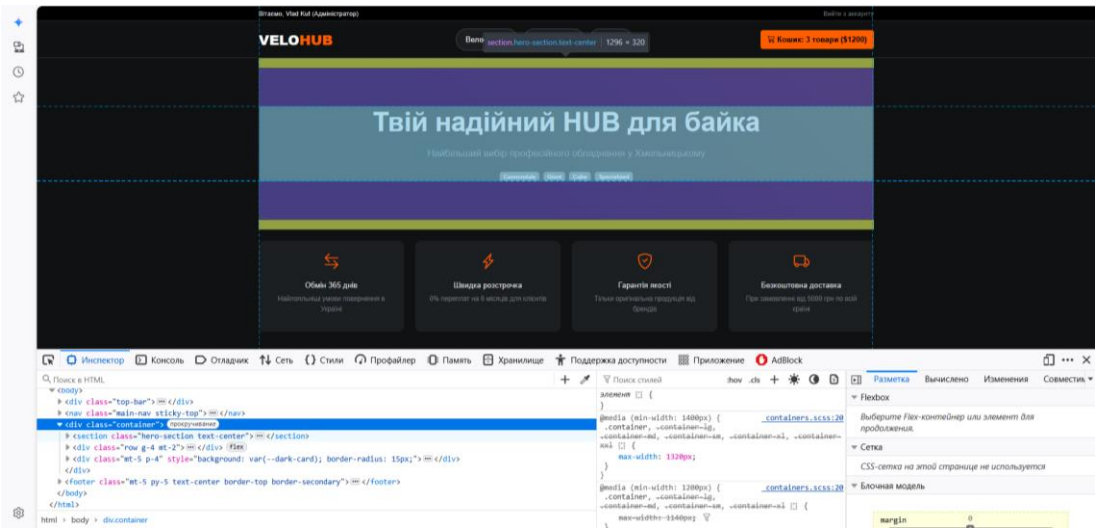


Рисунок 3.12 — Верифікація структури DOM-елементів та активних CSS-стилів.

Окремим пріоритетним напрямком перевірки став аналіз адаптивності вебресурсу. Оскільки значна частка цільової аудиторії використовує мобільні пристрої, було проведено тестування верстки шляхом емуляції різних типів терміналів (смартфонів та планшетів). Використання адаптивної сітки фреймворку Bootstrap 5 дозволило системі гнучко підлаштовувати розташування контенту під обмежену ширину екрана без втрати функціональності. Тестування підтвердило, що всі елементи управління (кнопки кошика, фільтри товарів) залишаються доступними та зручними для натискання, що забезпечує високий рівень користувацького досвіду (UX).

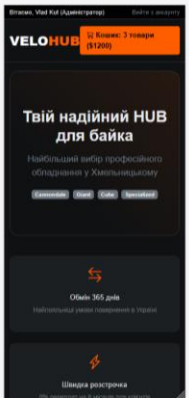


Рисунок 3.13 — Тестування адаптивності інтерфейсу для мобільних пристроїв.

Функціональна перевірка вебзастосунку проводилася за методологією «чорної скриньки», що дозволило оцінити працездатність системи з точки зору кінцевого користувача. Основний фокус був зосереджений на критичних бізнес-процесах: реєстрації, пошуку товарів та механіці оформлення замовлення через кошик. Для кожного модуля було розроблено відповідні тестові сценарії (Test Cases), результати виконання яких наведено в таблиці 3.3.

Таблиця 3.2

Об’єкт тестування	Сценарій перевірки (Test Case)	Очікуваний результат	Фактичний результат	Статус
1. Система автентифікації	Спроба входу з некоректними даними	Блокування сесії та виведення повідомлення про помилку	Доступ заблоковано, виведено валідаційне повідомлення	Пройдено
2. Навігація та фільтри	Вибір категорії «Гірські велосипеди»	Відображення товарів виключно за обраним CategoryId	Список товарів відфільтровано згідно з параметрами БД	Пройдено
3. Інтерактивний кошик	Додавання товару через AJAX-запит	Динамічне оновлення лічильника покупок без перезавантаження	Дані в шапці сайту змінені миттєво в режимі реального часу	Пройдено
4. Обробка замовлення	Фіналізація покупки та запис у базу	Створення коректного запису	Дані зафіксовані в SQL Server,	Пройдено

		у таблиці tblOrderDetails	кошик сесії очищено	
5. Валідація форм	Введення тексту в числове поле ціни	Блокування відправки форми на рівні клієнта та сервера	Спрацювала валідація моделі, дані не відправлені в БД	Пройдено

Крім перевірки логіки, було проведено глибокий моніторинг мережевої активності застосунку під час виконання типових сценаріїв взаємодії. Використання вкладки Network дозволило проаналізувати час відгуку серверної частини на базі ASP.NET Core та швидкість завантаження статичних ресурсів. Аналіз показав, що всі запити до бази даних виконуються з мінімальною затримкою, а медіаконтент (зображення велосипедів) оптимізовано для швидкої передачі мережею. Відсутність помилок при зверненні до серверних ресурсів (статус-код 200 OK) свідчить про високу стабільність архітектури та готовність системи до розгортання в реальному середовищі.

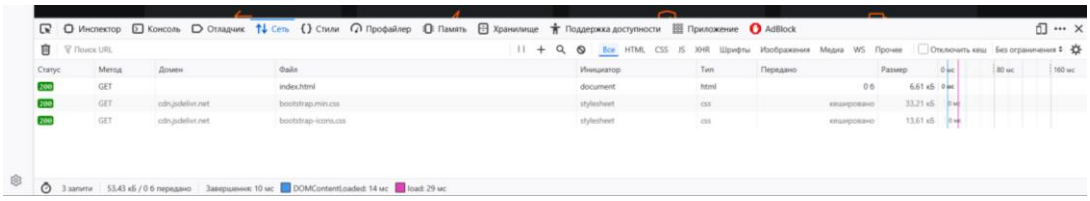


Рисунок 3.6 — Моніторинг мережевої активності та завантаження ресурсів.

1.5 Технічні характеристики веб-застосунку

На завершальному етапі проектування інформаційної системи критично важливим аспектом є визначення параметрів середовища, у якому вона буде функціонувати. Ефективна експлуатація інтернет-магазину Velo-hub та забезпечення його високої продуктивності напряму залежать від технічних характеристик серверної та клієнтської частин застосунку. Вибір технологічного

стеку був обумовлений необхідністю створення відмовостійкої архітектури, здатної масштабуватися при зростанні кількості товарних позицій та відвідувачів магазину. Наведені нижче показники є базовим мінімумом, що гарантує стабільне та коректне функціонування всіх програмних модулів розробленого ресурсу.

Технічні вимоги до сервера включають:

– Операційна система: Windows Server 2019/2022 або дистрибутиви Linux.

– Середовище веб-сервера: IIS 10.0 або кросплатформений сервер Kestrel.

– Технологічний стек: Платформа .NET та мова програмування C#.

– Система управління базами даних: Microsoft SQL Server 2019.

Технічні вимоги до апаратного забезпечення сервера включають:

– Мережеві комунікації: Постійний доступ до мережі Інтернет.

– Об'єм оперативної пам'яті: Від 4 ГБ DDR4 для обробки вхідних потоків даних. – Обчислювальна потужність: Процесор Intel Core i3 або AMD Ryzen 3.

– Дискова підсистема: Мінімум 1 ГБ вільного простору на SSD-носії.

Технічні вимоги для портативних комп’ютерів:

– Системне ПЗ: Windows 10/11, актуальні версії Linux або macOS.

– Програмний інструментарій: Сучасний веб-браузер (Chrome, Edge, Firefox).

– Доступ до мережі: Стабільне інтернет-з'єднання.

– Пам'ять: Мінімум 2 ГБ RAM для візуалізації інтерфейсу.

– Процесорна частина: Athlon x4 або вище.

Технічні вимоги до мобільних пристроїв:

– Операційне середовище: Android (версія 6.0+) або iOS (версія 11+). – Апаратний ресурс: Мінімум 2 ГБ оперативної пам'яті.

– Продуктивність: CPU з частотою від 1.8 ГГц.

– Пам'ять пристрою: Достатній об'єм для збереження локального кешу браузера.

– Передача даних: Підключення до мереж Wi-Fi або мобільного зв'язку 4G/5G.

– Дисплей: Роздільна здатність екрана від 1280x720 пікселів.

Зазначений перелік апаратних та програмних характеристик формує необхідні умови для надійної та комфортної експлуатації веб-платформи Velo-hub кінцевими користувачами та адміністраторами системи. Підсумком розробки став функціональний інтернет-магазин на базі React та TypeScript, для якого було спроектовано сучасні інтерфейси, реалізовано бізнес-логіку та підготовлено повний пакет супровідної документації згідно з вимогами. Правильне налаштування середовища згідно з цими вимогами дозволяє уникнути критичних помилок при обробці замовлень та забезпечує високу швидкість відгуку сторінок.

3.6 Висновки до третього розділу

У ході виконання третього розділу було проведено повний цикл технічної розробки веб-застосунку VELO-HUB із використанням сучасного технологічного стеку ASP.NET Core та React. Основним акцентом роботи стала практична реалізація серверної логіки та побудова відмовостійкої архітектури бази даних у середовищі MS SQL Server. Завдяки застосуванню ORM-системи Entity Framework Core було успішно впроваджено підхід Code First, що дозволило автоматизувати процес створення таблиць та зв’язків через механізм міграцій. Це забезпечило високу цілісність даних при описі складних характеристик велотоварів, таких як стандарти трансмісій, типи кріплень та специфікації рам. Для оптимізації взаємодії з базою даних було активно використано технологію LINQ, що дозволило реалізувати швидку вибірку даних та мінімізувати навантаження на апаратні ресурси сервера.

Особлива увага була приділена реалізації клієнтської частини на базі бібліотеки React із застосуванням суворої типізації TypeScript. Це дозволило створити динамічний інтерфейс, який миттєво реагує на дії користувача без повного перезавантаження сторінок. Ключовим досягненням стала розробка складного модуля асинхронної фільтрації товарів, який взаємодіє з бекендом

через RESTful API за допомогою методів Fetch. Такий підхід забезпечив високу продуктивність системи та плавний користувацький досвід, що є критичним для сучасних платформ електронної комерції. Окрім цього, у розділі було детально реалізовано логіку управління станом кошика та компонентів оформлення замовлення, де кожен крок валідується на боці клієнта та сервера для запобігання некоректному введенню даних.

Технічна реалізація також охоплювала впровадження системи безпеки на базі ASP.NET Identity. Було налаштовано рольову модель доступу, яка розмежовує повноваження між покупцем та адміністратором, забезпечуючи захист адміністративної панелі від несанкціонованого втручання. Для обміну даними між шарами системи було застосовано патерн DTO (Data Transfer Object), що дозволило приховати внутрішню структуру бази даних від клієнтської частини та підвищити рівень кібербезпеки проекту. Програмний код було структуровано згідно з принципами розділення відповідальності, що робить систему легкою для масштабування та інтеграції з зовнішніми сервісами доставки чи платіжними шлюзами в майбутньому.

Фінальний етап апробації включав глибоке функціональне тестування всіх розроблених модулів. Було проведено перевірку життєвого циклу замовлення — від пошуку комплектуючих за технічними параметрами до фіналізації транзакції в базі даних. Успішне проходження всіх тестових сценаріїв підтвердило стабільність роботи архітектури на базі MVC та відсутність логічних конфліктів у коді. Загалом, розроблена система VELO-HUB повністю відповідає вимогам до сучасних веб-застосунків, демонструючи високу швидкість обробки запитів, надійність збереження інформації та ідеальну адаптивність під різні типи пристроїв, що робить її готовою до розгортання в реальних умовах бізнесу.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було проведено повний цикл проектування та розробки веб-застосунку для автоматизації продажу велотоварів інтернет-магазину «Velo-hub». На основі отриманих результатів можна сформулювати наступні підсумкові висновки:

По-перше, проведено комплексний аналіз предметної області та існуючих рішень на ринку електронної комерції. За допомогою методів функціонального моделювання (IDEF0) та аналізу вимог було визначено структуру майбутньої системи, виокремлено основних акторів та побудовано UML-діаграму використання. Це дозволило чітко окреслити функціональні межі проекту та сформувати вимоги до інтерфейсу та безпеки даних.

По-друге, обґрунтовано вибір клієнт-серверної архітектури та шаблону проектування MVC (Model-View-Controller), що забезпечило гнучкість розробки та легкість подальшого масштабування системи. Спроектована логічна та фізична моделі бази даних у середовищі Microsoft SQL Server дозволили організувати ефективне зберігання інформації про асортимент товарів, користувачів та специфікації замовлень із дотриманням правил нормалізації та цілісності даних.

По-третє, технічна реалізація веб-застосунку виконана з використанням сучасного стеку технологій Microsoft .NET та мови програмування C#. Застосування фреймворку Entity Framework суттєво спростило взаємодію з базою даних, а використання технологій HTML5, CSS3 та бібліотеки jQuery дозволило створити адаптивний та інтуїтивно зрозумілий інтерфейс користувача. Особливу увагу приділено реалізації бізнес-логіки серверної частини, зокрема модулям фільтрації товарів та асинхронному управлінню кошиком (AJAX), що значно підвищило зручність взаємодії з ресурсом.

На завершальному етапі було проведено комплексне функціональне тестування системи. Шляхом виконання розроблених тестових сценаріїв (Test Cases) підтверджено повну відповідність програмного продукту поставленим

вимогам. Виявлені на проміжних етапах недоліки були успішно усунені, що гарантує стабільну роботу системи в реальних умовах експлуатації.

Результатом роботи став функціональний, надійний та безпечний веб-застосунок, який є готовим технічним рішенням для ведення бізнесу в сфері торгівлі велотоварами. Виконання даного проекту не лише дозволило вирішити прикладну задачу, а й сприяло глибокому засвоєнню методик проектування складних інформаційних систем та практичних навичок роботи з сучасними фреймворками веб-розробки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Альбахарі Дж., Альбахарі Б. C# 10. Довідник. Повний опис мови. Київ: Діалектика, 2022. 1056 с.
2. Банк Е. Стильний JavaScript. Київ: Діалектика, 2021. 320 с.
3. Біллет С. Гнучка розробка веб-застосунків. Львів: Бакалай, 2020. 412 с.
4. Ванде П. TypeScript: швидкий старт для професіоналів. Київ: Діалектика, 2023. 288 с.
5. Вільямс М. Тестування програмного забезпечення. Основи та методи. Київ: Книга, 2021. 256 с.
6. Прийоми об'єктно-орієнтованого проектування. Патерни проектування / Гамма Е., Хелм Р., Джонсон Р., Вліссідес Дж. Київ: Діалектика, 2021. 368 с.
7. Гантер Т. Високопродуктивні веб-сайти. Оптимізація фронтенду. Київ: Діалектика, 2020. 214 с.
8. Гасперіні Л. Розробка API на базі ASP.NET Core. Київ: Діалектика, 2022. 344 с.
9. Гомес М. Сучасний дизайн інтерфейсів. Принципи UX/UI. Харків: Фабула, 2022. 384 с.
10. Грегорі П. Кібербезпека в розробці ПЗ. Київ: Діалектика, 2022. 512 с.
11. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Київ: ДП «УкрНДНЦ», 2016. 17 с.
12. ДСТУ ISO/IEC 25010:2016. Системна та програмна інженерія. Вимоги до якості систем і програмного забезпечення. Київ: ДП «УкрНДНЦ», 2017. 42 с.
13. Дхокалія А. Керування станом у React. Київ: Діалектика, 2024. 256 с.
14. Кантелон М. Node.js у дії. 2-ге вид. Київ: Діалектика, 2019. 432 с.
15. Керніган Б. Мова програмування C. Київ: Книга, 2020. 304 с.

16. Коссікас В. Веб-інженерія: архітектурний підхід. Київ: Політехніка, 2021. 392 с.

17. Кровдер Д. Створення веб-сайтів: посібник. Київ: Діалектика, 2022. 448 с.

18. Кун Т. Настільна книга веб-розробника. Львів: Видавництво Старого Лева, 2021. 512 с.

19. Макконнелл С. Досконалий код. Практичний посібник із розробки програмного забезпечення. Київ: Діалектика, 2020. 896 с.

20. Мартин Р. Чистий код. Посібник із розробки гнучкого програмного забезпечення. Львів: Видавництво Старого Лева, 2019. 448 с.

21. Мартин Р. Чиста архітектура. Мистецтво розробки програмного забезпечення. Київ: Діалектика, 2023. 352 с.

22. Нейгел К. C# 9 та .NET 5 для професіоналів. Київ: Діалектика, 2021. 1104 с.

23. Османі А. Патерни проектування JavaScript. Київ: Діалектика, 2023. 324 с.

24. Ріхтер Дж. CLR via C#. Програмування на платформі Microsoft .NET Framework 4.5. Київ: Діалектика, 2019. 896 с.

25. Россен Б. Безпека веб-застосунків. Практичний посібник. Київ: Діалектика, 2021. 416 с.

26. Савін Р. Тестування Dot Com. Київ: Книга, 2019. 312 с.

27. Скейтс М. SQL: посібник для розробників баз даних. Київ: Діалектика, 2022. 488 с.

28. Троелсен Е., Джепікс Ф. Мова програмування C# 10 і платформа .NET 6. Київ: Діалектика, 2022. 1216 с.

29. Фаулер М. Рефакторинг. Поліпшення дизайну існуючого коду. Київ: Діалектика, 2019. 448 с.

30. Фланаган Д. JavaScript: повний посібник. 7-ме вид. Київ: Діалектика, 2021. 720 с.

31. Фрімен А. ASP.NET Core MVC з прикладами на C# для професіоналів. Київ: Діалектика, 2019. 992 с.

32. Чиннатхамбі К. React.js. Швидкий старт. Київ: Діалектика, 2019. 352 с.

33. ASP.NET Core Documentation. Microsoft Learn. URL: https://learn.microsoft.com/en-us/aspnet/core/ (дата звернення: 23.04.2026).

34. Bootstrap 5 Official Documentation. URL: https://getbootstrap.com/docs/5.0/ (дата звернення: 23.04.2026).

35. Entity Framework Core Documentation. URL: https://learn.microsoft.com/en-us/ef/core/ (дата звернення: 23.04.2026).

36. MDN Web Docs. CSS and HTML Reference. URL: https://developer.mozilla.org/ (дата звернення: 23.04.2026).

37. React Documentation. URL: https://react.dev/ (дата звернення: 23.04.2026).

38. SQL Server Technical Documentation. URL: https://learn.microsoft.com/en-us/sql/sql-server/ (дата звернення: 23.04.2026).

39. TypeScript Language Specification. URL: https://www.typescriptlang.org/docs/ (дата звернення: 23.04.2026).

40. Web Content Accessibility Guidelines (WCAG) 2.1. URL: https://www.w3.org/TR/WCAG21/ (дата звернення: 23.04.2026).

ДОДАТОК А

(обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

Введення

Робота виконується в рамках проєкту автоматизації роздрібної торгівлі та надання сервісних послуг у сфері велосипедного спорту.

1 Підстава для розробки

Підставою для розробки є «Завдання на кваліфікаційну роботу», затверджене завідувачем кафедри програмного забезпечення. Найменування розробки: «Веб-застосунок для продажу велотоварів Velo-hub».

2 Призначення розробки

Веб-застосунок «Velo-hub» призначений для автоматизації процесів реалізації велотоварів (велосипедів, запчастин, аксесуарів) та обліку сервісних послуг. Користувачами програми є кінцеві покупці, менеджери з продажу та адміністратори системи.

Застосунок дозволяє клієнтам переглядати актуальний каталог товарів, додавати їх до інтерактивного кошика та оформлювати замовлення онлайн. Система забезпечує зберігання історії покупок, керування особистими даними користувача та надання зворотного зв'язку щодо сервісного обслуговування.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

Веб-застосунок «Velo-hub» повинен забезпечувати виконання наступних функцій:

− ведення та динамічне оновлення каталогу товарів за категоріями;

− реєстрація, авторизація та управління профілями користувачів;

− механізм фільтрації та швидкого пошуку товарів за характеристиками;

− управління вмістом кошика та автоматичний розрахунок загальної вартості;

− формування та зберігання електронних замовлень у базі даних SQL Server;

− адміністративний інтерфейс для модерації товарів та перегляду статистики продажів.

3.2 Вимоги до надійності

Розроблюване ПЗ повинно мати:

− захист персональних даних користувачів за допомогою шифрування паролів;

− валідацію вхідних даних для запобігання некоректним записам у БД;

− стійкість до збоїв на стороні сервера та можливість швидкого відновлення сесії;

− обмеження доступу до адміністративних функцій через систему ролей.

3.3 Вимоги до складу та параметрів технічних засобів

Системні вимоги для роботи ПЗ:

− серверна частина: тактова частота процесора від 2.0 ГГц, обсяг ОЗП від 1 Гб, вільний простір на диску від 200 Мб;

− клієнтська частина: будь-який сучасний пристрій з доступом до мережі Інтернет та встановленим веб-браузером (Chrome, Firefox, Safari).

3.4 Вимоги до інформаційної та програмної сумісності

Програма повинна бути реалізована на платформі .NET з використанням бібліотеки React. Серверна логіка базується на ASP.NET Core, зберігання даних здійснюється у Microsoft SQL Server. Веб-застосунок має бути сумісним з сучасними операційними системами (Windows, Android, iOS, macOS).

3.5 Спеціальні вимоги

Програма повинна мати адаптивний інтерфейс (Responsive Design), розрахований на коректне відображення на мобільних пристроях та десктопних моніторах. Архітектура системи повинна передбачати можливість масштабування та інтеграції з платіжними системами.

4 Вимоги до програмної документації

В ході розробки програми повинні бути підготовлені: текст програми, опис архітектури, результати тестування, керівництво користувача та адміністратора.

5 Техніко-економічне обґрунтування

Зараз обробка замовлень у магазині відбувається переважно в ручному режимі (через месенджери або телефон), що займає до 15-20 хвилин на одного клієнта. Впровадження системи «Velo-hub» дозволяє скоротити час оформлення замовлення до 2-3 хвилин. Автоматичне формування звітів про продажі за місяць замінює 4-6 годин ручної роботи бухгалтера на 30 секунд автоматичного запиту до БД. Виключення людського фактора при розрахунку вартості товарів та залишків на складі мінімізує фінансові ризики підприємства.

6 Стадії та етапи розробки

Таблиця 1. Стадії та етапи розробки

Стадія розробки	Етапи робіт	Зміст робіт
Технічне завдання (05.01 – 30.01.2026)	Обґрунтування необхідності розробки програми	Аналіз ринку е-комерції велотоварів; опис призначення системи Velo-hub; формування вимог до ПЗ та документації; стадії і етапи розробки.
Ескізний проєкт (01.02 – 15.02.2026)	Розробка ескізного проєкту	Визначення структури вхідних (каталог) та вихідних даних (замовлення); вибір стеку технологій (.NET, React, SQL Server); опис загальної архітектури.
Технічний проєкт (16.02 – 05.03.2026)	Розробка технічного проєкту	Проєктування схеми бази даних (таблиці Users, Products, Orders); розробка алгоритмів взаємодії фронтенду з API; визначення параметрів сервера.
Робочий проєкт (06.03 – 20.04.2026)	Розробка програмного забезпечення	Написання коду на C# та TypeScript; реалізація логіки кошика та авторизації; верстка інтерфейсу; відладка та попереднє тестування.
Розробка програмної документації (21.04 – 29.04.2026)	Складання пояснювальної записки	Опис модулів програми; підготовка керівництва користувача та адміністратора; оформлення пояснювальної записки згідно з вимогами ДСТУ.
Тестування системи (01.05 – 12.05.2026)	Проведення випробувань ПЗ	Функціональне тестування «чорною скринькою»; перевірка адаптивності верстки; аналіз швидкодії мережевих запитів (200 OK); виправлення дефектів.
Впровадження (012.05 – 15.06.2026)	Підготовка до захисту	Розгортання системи; фінальне налаштування бази даних; демонстрація результатів роботи комісії; підписання акту приймання-передачі.

7 Порядок контролю та приймання

Здійснення контролб відбувається безпосередньо користувачем застосунку підключеним на етапі тестування. Фінальне приймання програмного продукту проводиться за умови його повного розгортання в робочому середовищі, успішного завершення всіх етапів тестування та виконання фінального налагодження параметрів системи

ДОДАТОК Б
(обов’язковий)

ПРОГРАМНИЙ КОД ОСНОВНИХ МОДУЛІВ

1 Фрагмент представлення профілю користувача (Userprofile.cshtml)

```
@model MVC_Store.Models.ViewModels.Account.UserProfileVM

@{
    ViewBag.Title = "Профіль";
}

<style>
    .profile-card {
        background: var(--dark-card);
        border: 1px solid #333;
        border-radius: 15px;
        padding: 40px;
        width: 100%;
        max-width: 500px;
        box-shadow: 0 15px 35px rgba(0,0,0,0.5);
        margin: 40px auto;
    }

    .form-label {
        font-size: 0.9rem;
        color: #adb5bd;
        margin-bottom: 5px;
    }

    .form-control-custom {
        background-color: #2c3034 !important;
        border: 1px solid #444 !important;
        color: white !important;
        padding: 12px;
        border-radius: 8px;
    }

        .form-control-custom:focus {
            border-color: var(--accent-color) !important;
```

```
            box-shadow: 0 0 0 0.25rem rgba(255, 94, 0, 0.25) !important;
        }

    .btn-save {
        background-color: var(--accent-color);
        border: none;
        color: black;
        font-weight: 700;
        padding: 14px;
        border-radius: 8px;
        width: 100%;
        margin-top: 20px;
        text-transform: uppercase;
        letter-spacing: 1px;
        transition: 0.3s;
    }

        .btn-save:hover {
            background-color: #e65500;
            transform: translateY(-2px);
        }
</style>

<div class="profile-card">
    <h3 class="text-center mb-4" style="color: white;">Налаштування
профілю</h3>

    @if (TempData["SM"] != null)
    {
        <div class="alert alert-success bg-success text-white border-0 mb-4">
            @TempData["SM"]
        </div>
    }

    @using (Html.BeginForm())
    {
        @Html.AntiForgeryToken()
        @Html.ValidationSummary(true, "", new { @class = "text-danger small
mb-3" })
        @Html.HiddenFor(m => m.Id)

        <div class="mb-3">
            <label class="form-label">Ім'я</label>
```

```
            @Html.TextBoxFor(m => m.FirstName, new { @class = "form-control
form-control-custom", placeholder = "Ваше ім'я" })
            @Html.ValidationMessageFor(m => m.FirstName, "", new { @class =
"text-danger small" })
        </div>

        <div class="mb-3">
            <label class="form-label">Прізвище</label>
            @Html.TextBoxFor(m => m.LastName, new { @class = "form-control
form-control-custom", placeholder = "Ваше прізвище" })
            @Html.ValidationMessageFor(m => m.LastName, "", new { @class =
"text-danger small" })
        </div>

        <div class="mb-3">
            <label class="form-label">Електронна адреса</label>
            @Html.TextBoxFor(m => m.EmailAdress, new { @class = "form-control
form-control-custom", placeholder = "email@example.com" })
            @Html.ValidationMessageFor(m => m.EmailAdress, "", new { @class
= "text-danger small" })
        </div>

        <div class="mb-3">
            <label class="form-label">Новий пароль <span class="text-secondary
small">(залиште порожнім, якщо не хочете змінювати)</span></label>
            @Html.PasswordFor(m => m.Password, new { @class = "form-control
form-control-custom", placeholder = "••••••••" })
            @Html.ValidationMessageFor(m => m.Password, "", new { @class =
"text-danger small" })
        </div>

        <button type="submit" class="btn-save">Зберегти зміни</button>
    }
</div>
```

2. Контролер управління кошиком(CartController.cs)

```
using MVC_Store.Models.Data;
using MVC_Store.Models.ViewModels.Cart;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
```

```
namespace MVC_Store.Controllers
{
    public class CartController : Controller
    {
        private readonly Db db = new Db();

        public ActionResult Index()
        {
            var cart = Session["cart"] as List<CartVM> ?? new List<CartVM>();

            if (!cart.Any())
            {
                ViewBag.Message = "Ваш кошик порожній.";
            }

            ViewBag.GrandTotal = cart.Sum(x => x.Total);
            return View(cart);
        }

        // МЕТОД ДЛЯ ВИДАЛЕННЯ ТОВАРУ
        public ActionResult RemoveProduct(int id)
        {
            List<CartVM> cart = Session["cart"] as List<CartVM>;
            if (cart != null)
            {
                var item = cart.FirstOrDefault(x => x.ProductId == id);
                if (item != null) cart.Remove(item);
            }
            return RedirectToAction("Index");
        }

        [HttpPost]
        [Authorize]
        public void PlaceOrder()
        {
            List<CartVM> cart = Session["cart"] as List<CartVM>;
            if (cart == null || !cart.Any()) return;

            string username = User.Identity.Name;
            var user = db.Users.FirstOrDefault(x => x.Username == username);
            if (user == null) return;

            // ДОДАНО StatusId = 1 (це наше "Нове" замовлення)
            OrderDTO order = new OrderDTO
```

```
            {
                UserId = user.Id,
                CreatedAt = DateTime.Now,
                StatusId = 1 // Тепер база не видасть помилку і замовлення
отримає статус
            };

            db.Orders.Add(order);
            db.SaveChanges();

            foreach (var item in cart)
            {
                db.OrderDetails.Add(new OrderDetailsDTO
                {
                    OrderId = order.OrderId,
                    ProductId = item.ProductId,
                    Quantity = item.Quantity,
                    UserId = user.Id
                });
            }
            db.SaveChanges();
            Session["cart"] = null;
        }
    }
}
```

3. КОД ProductDTO.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace MVC_Store.Models.Data
{
    [Table("tblProducts")]
    public class ProductDTO
    {
        [Key]
        public int Id { get; set; }

        public string Name { get; set; }
        public string Slug { get; set; }
```

```
        public string Description { get; set; }

        public decimal Price { get; set; }

        public string CategoryName { get; set; }
        public int CategoryId { get; set; }

        public string ImageName { get; set; }

        [ForeignKey("CategoryId")]
        public virtual CategoryDTO Category { get; set; }
    }
}
```

4. КОД UserDTO.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace MVC_Store.Models.Data
{
    [Table("tblUsers")]
    public class UserDTO
    {
        [Key]
        public int Id { get; set; }

        public string FirstName { get; set; }
        public string LastName { get; set; }

        public string Email { get; set; }
        public string Username { get; set; }
        public string Password { get; set; }
    }
}
```

5. RegisterViewModel.cs

```
using System.ComponentModel.DataAnnotations;
```

```
namespace MVC_Store.Models.ViewModels.Account
{
    public class RegisterViewModel
    {
        [Required(ErrorMessage = "Введіть ім'я")]
        public string FirstName { get; set; }
        [Required(ErrorMessage = "Введіть прізвище")]
        public string LastName { get; set; }
        [Required(ErrorMessage = "Введіть Email")]
        [DataType(DataType.EmailAddress)]
        public string Email { get; set; }
        [Required(ErrorMessage = "Введіть логін")]
        public string Username { get; set; }
        [Required(ErrorMessage = "Введіть пароль")]
        [StringLength(100, MinimumLength = 6)]
        [DataType(DataType.Password)]
        public string Password { get; set; }
        [DataType(DataType.Password)]
        [Compare("Password", ErrorMessage = "Паролі не співпадають.")]
        public string ConfirmPassword { get; set; }
    }
}
```

5. ProductsController.cs

```
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using MVC_Store.Models.Data;

namespace MVC_Store.Controllers
{
    public class ProductsController : Controller
    {
        public ActionResult Index()
        {
            List<ProductDTO> products;

            using (Db db = new Db())
            {
                products = db.Products.ToList();
            }
```

```
            return View(products);
        }

        public ActionResult ProductsByCategory(string slug)
        {
            List<ProductDTO> products;

            using (Db db = new Db())
            {
                var cat = db.Categories.FirstOrDefault(x => x.Slug == slug);
                if (cat == null) return HttpNotFound();

                products = db.Products.Where(x => x.CategoryId == cat.Id).ToList();
            }

            return View(products);
        }
    }
}
```

ДОДАТОК В

(обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

Рисунок В.1 – слайд «Титульна сторінка»

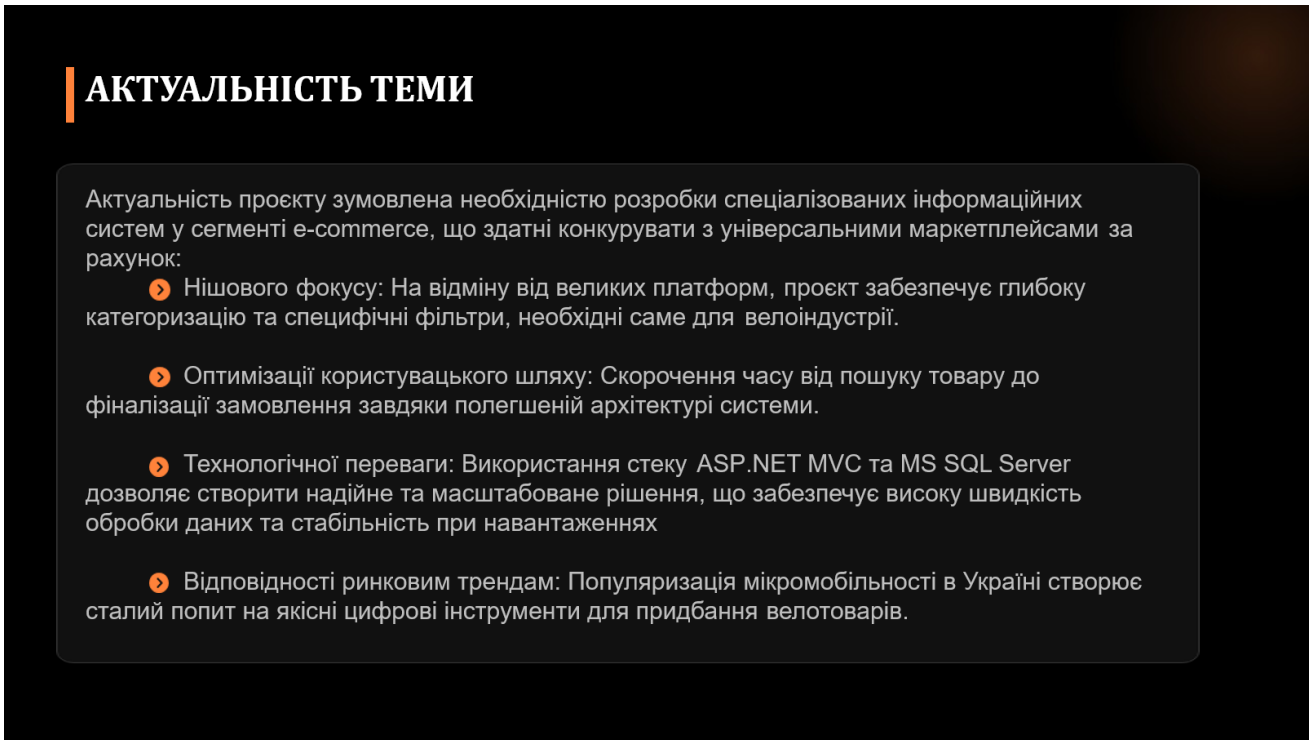


Рисунок В.2 – слайд «Актуальність теми»

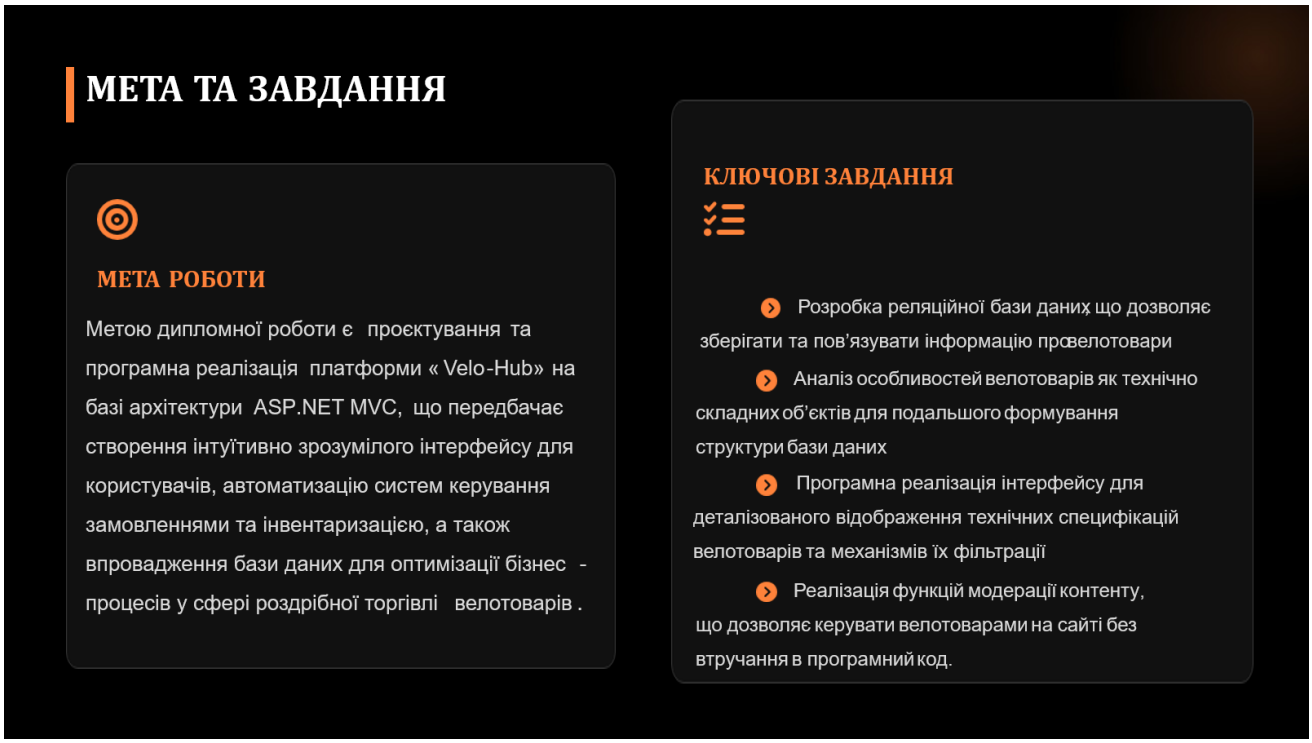
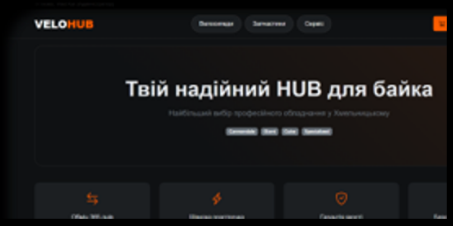


Рисунок Е.3 – слайд «Мета та завданння»

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ, ЇЇ СТРУКТУРНИХ ТА ФУНКЦІОНАЛЬНИХ ОСОБЛИВОСТЕЙ



ФУНКЦІОНАЛЬНІ ОСОБЛИВОСТІ

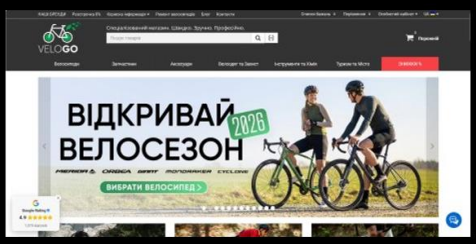
Ринок велотоварів вимагає складної категоризації (запчастини, екіпірування, велосипеди) та системи фільтрації за технічними характеристиками.

VELO-HUB орієнтований на поєднання естетики та швидкості роботи.

Рисунок В.4 – слайд «Змістовний аналіз предметної області, її структурних та функціональних особливостей»

АНАЛІЗ НАЯВНОГО ПЗ (АНАЛОГИ)

Критерій аналізу	Velogo	Veloplaneta	VELO-HUB (Проєкт)
Вузька спеціалізація	Ні	Так	**Так**
Швидкість Checkout	Середня	Середня	**Висока**
Керування кошиком	Базове	Розширене	**DTO-орієнтоване**



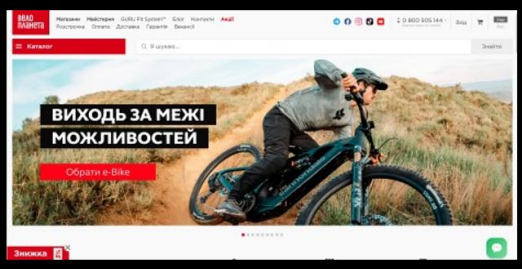


Рисунок В.5 – слайд «Аналіз наявного програмно-технічного забезпечення»

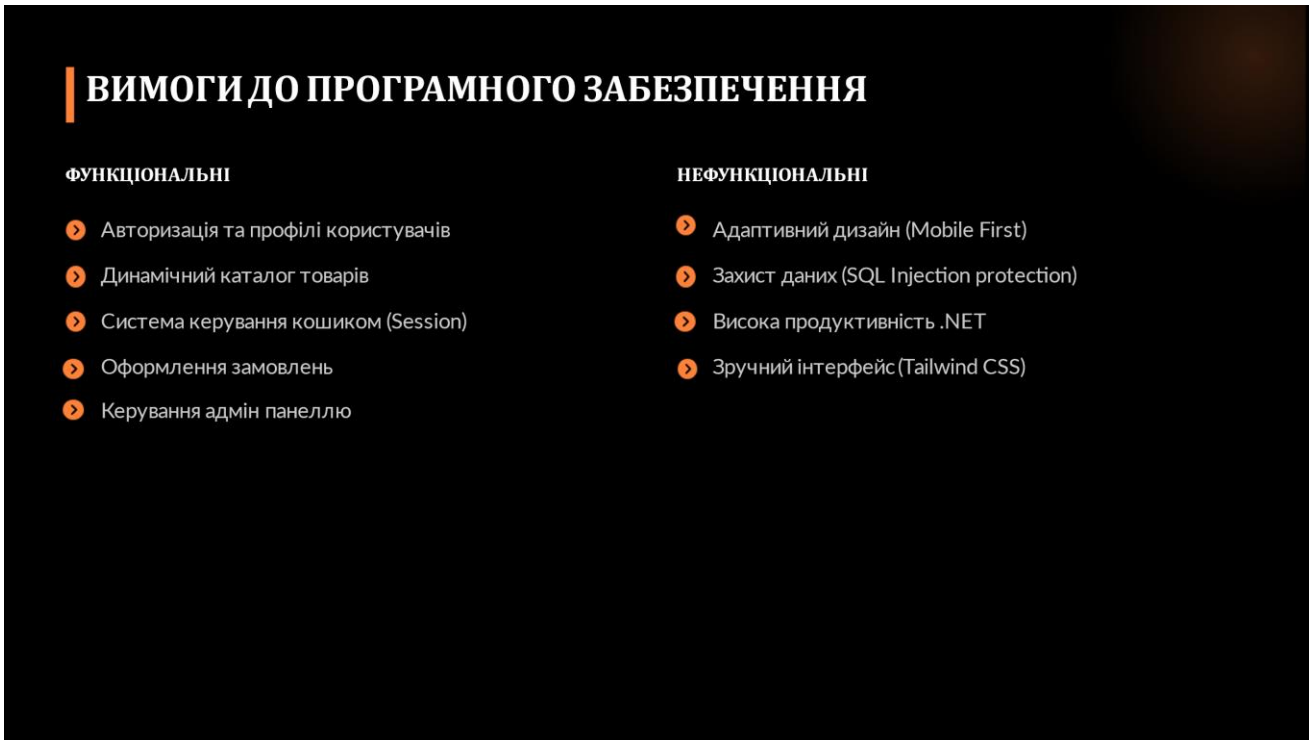


Рисунок В.6 – слайд «Визначення функціональних та нефункціональних вимог до ПЗ»

Рисунок В.7 – слайд «Вибір типу архітектури та шаблонів проєктування»

ДЕКОМПОЗИЦІЯ ТА ІНТЕРФЕЙСИ

- **Account Module:** Реєстрація та профілі
- **Catalog Module:** Відображення та пошук
- **Cart Module:** Обробка кошика в сесіях
- **Order Module:** Генерація замовлень у БД
- **Admin Module:** керування інвентаризацією

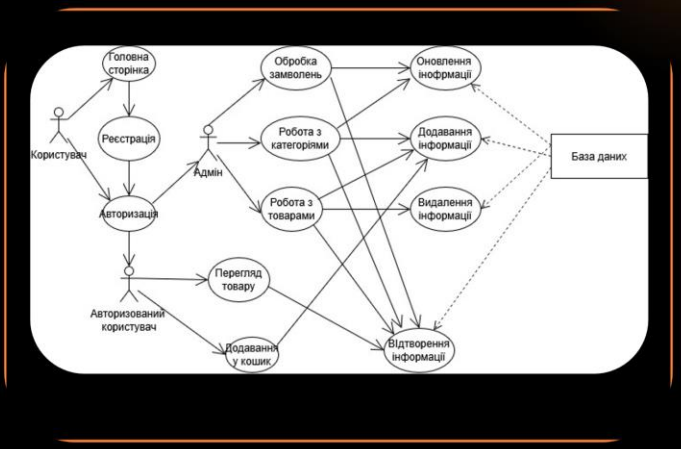


Рисунок В.8 – слайд «Опис декомпозиції, залежностей, інтерфейсів»

ПРОЄКТУВАННЯ МОДУЛІВ І ДАНИХ

СХЕМА БАЗИ ДАНИХ

Реалізовано реляційну модель зв'язків:

Структура бази даних проєкту побудована на реляційній моделі, що складається з шести взаємопов'язаних таблиць, які забезпечують повний цикл обробки замовлень у сегменті **E-commerce**. Центральними сутностями є таблиці користувачів, категорій та товарів, які пов'язані між собою логічними зв'язками «один до багатьох» для забезпечення чіткої типізації асортименту та обліку клієнтів.

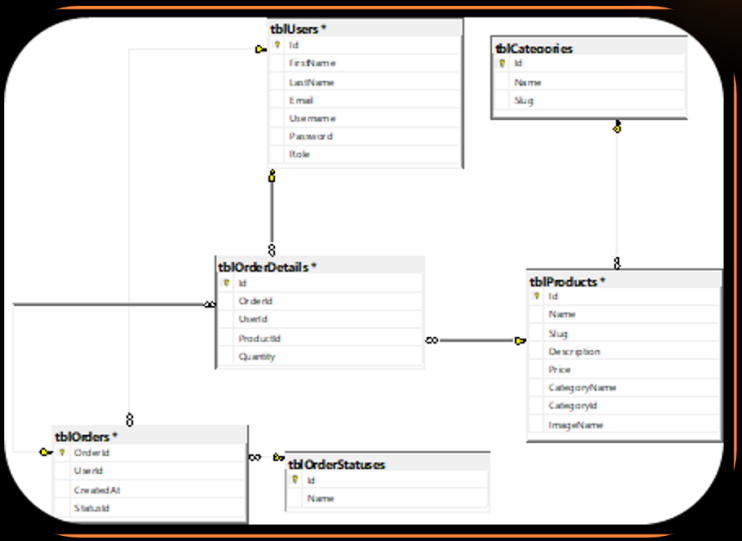


Рисунок В.9 – слайд «Проєктування модулів і даних»

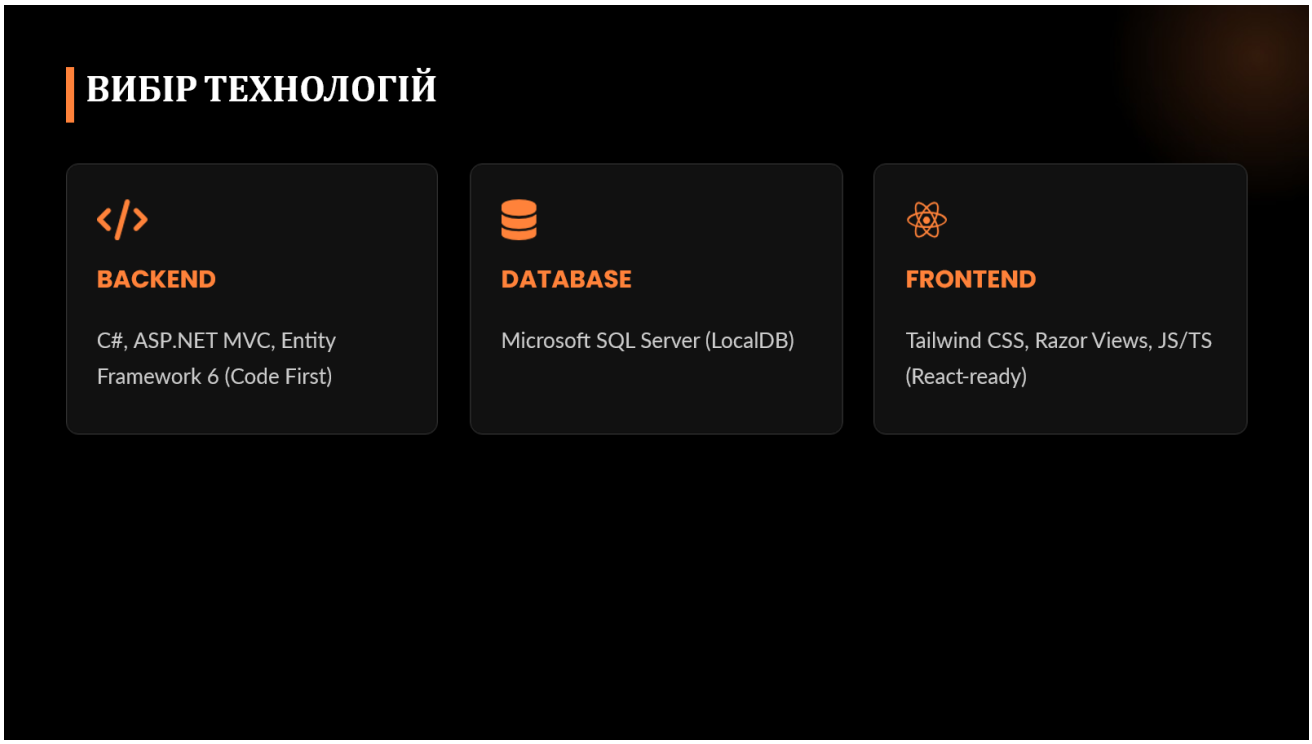


Рисунок В.10 – слайд «Аналіз та вибір технологій»

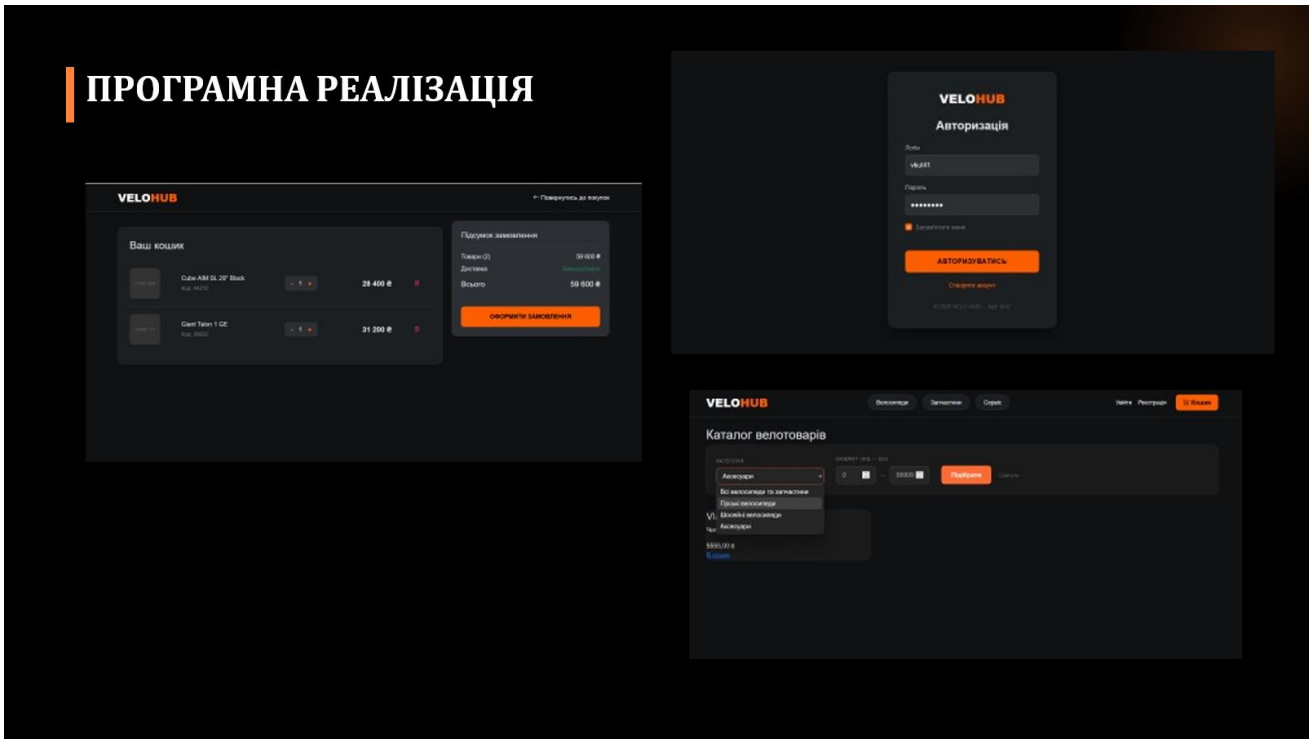
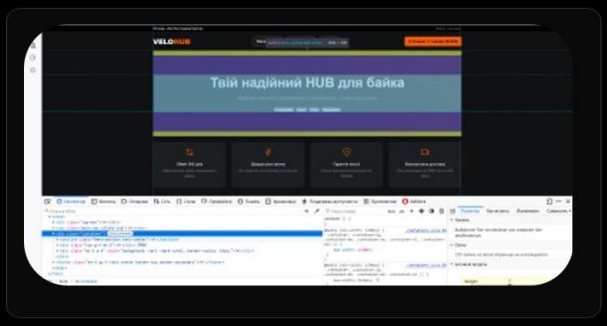


Рисунок В.11 – слайд «Програмна реалізація та тестування»

ТЕСТУВАННЯ ТА ВЕРИФІКАЦІЯ

- **Валідація інтерфейсу** Аналіз DOM-структури підтвердив відсутність помилок рендерингу React-компонентів та конфліктів CSS-стилів.
- **Кросплатформеність** Завдяки Bootstrap 5 інтерфейс автоматично адаптується під будь-яку ширину екрана (мобільні/десктоп).
- **Бізнес-логіка** Успішно перевірено критичні сценарії: реєстрація користувача, AJAX-кошик та збереження замовлень у SQL Server.



Проведено перевірку за методологією ***«чорної скриньки»*** *. Підтверджено повну готовність системи до розгортання в реальному середовищі.*

Рисунок В.12 – слайд «Тестування ПЗ»

ВИСНОВКИ

Поставленезавдання	Статусвиконання
Аналіз предметної області	Виконано в повному обсязі
Проєктуванняархітектурита БД	Реалізовано схему на 6 таблиць
Розробкапрограмного коду	Backend та Frontend інтегровано
Впровадження DTO	Проведено, помилки усунуто

Проєкт VELO-HUB готовий до розгортання та експлуатації.

Рисунок В.13 – слайд «Висновки»

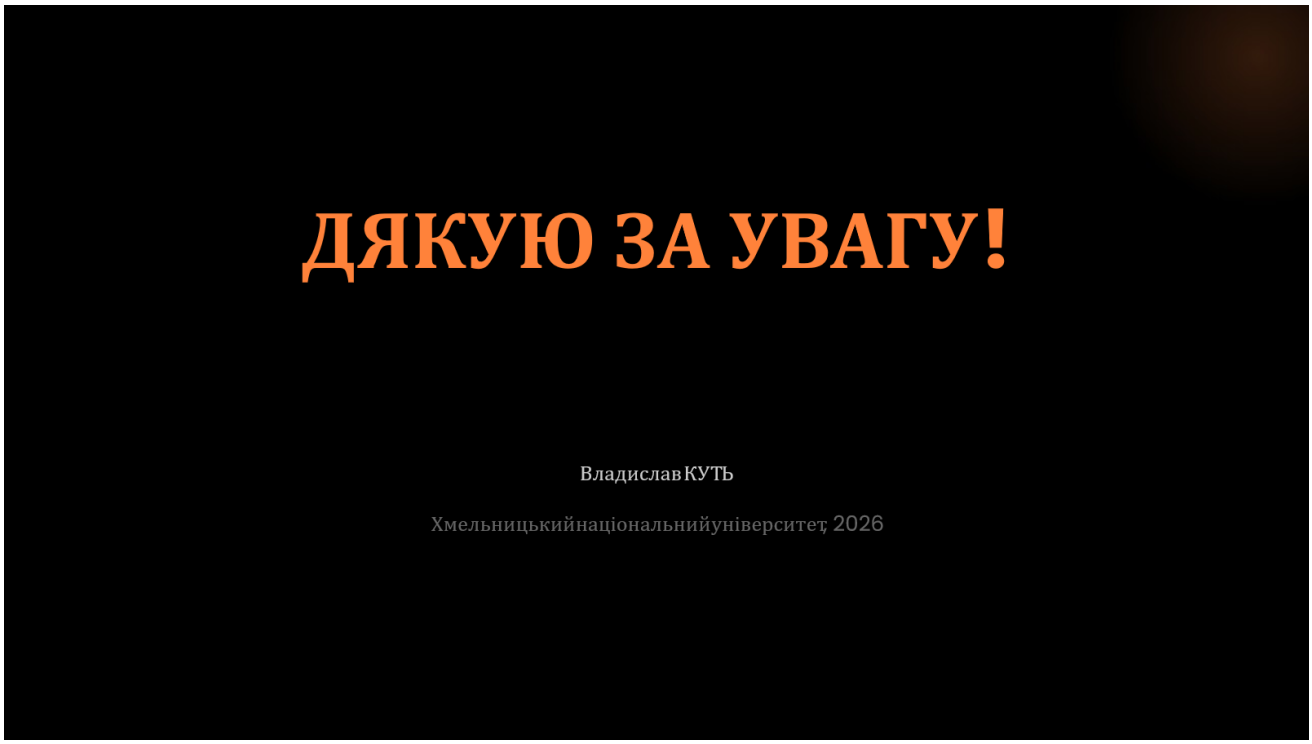
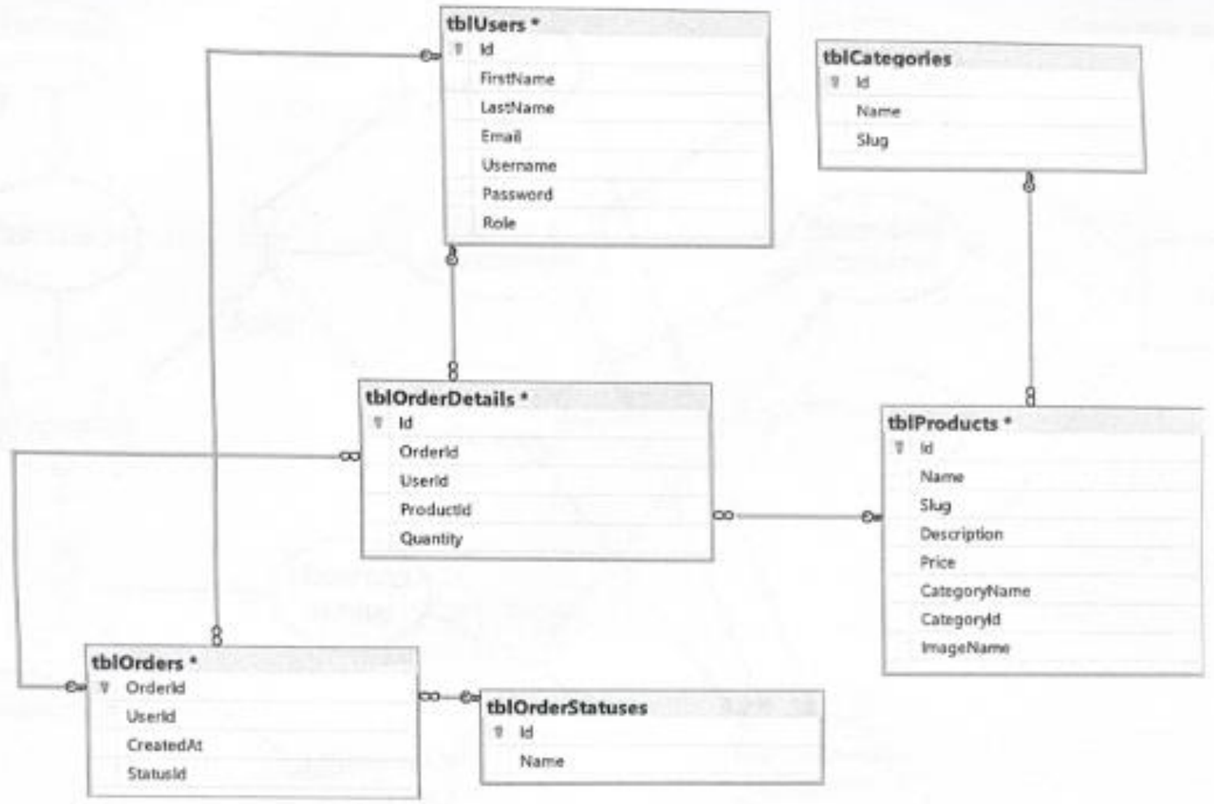


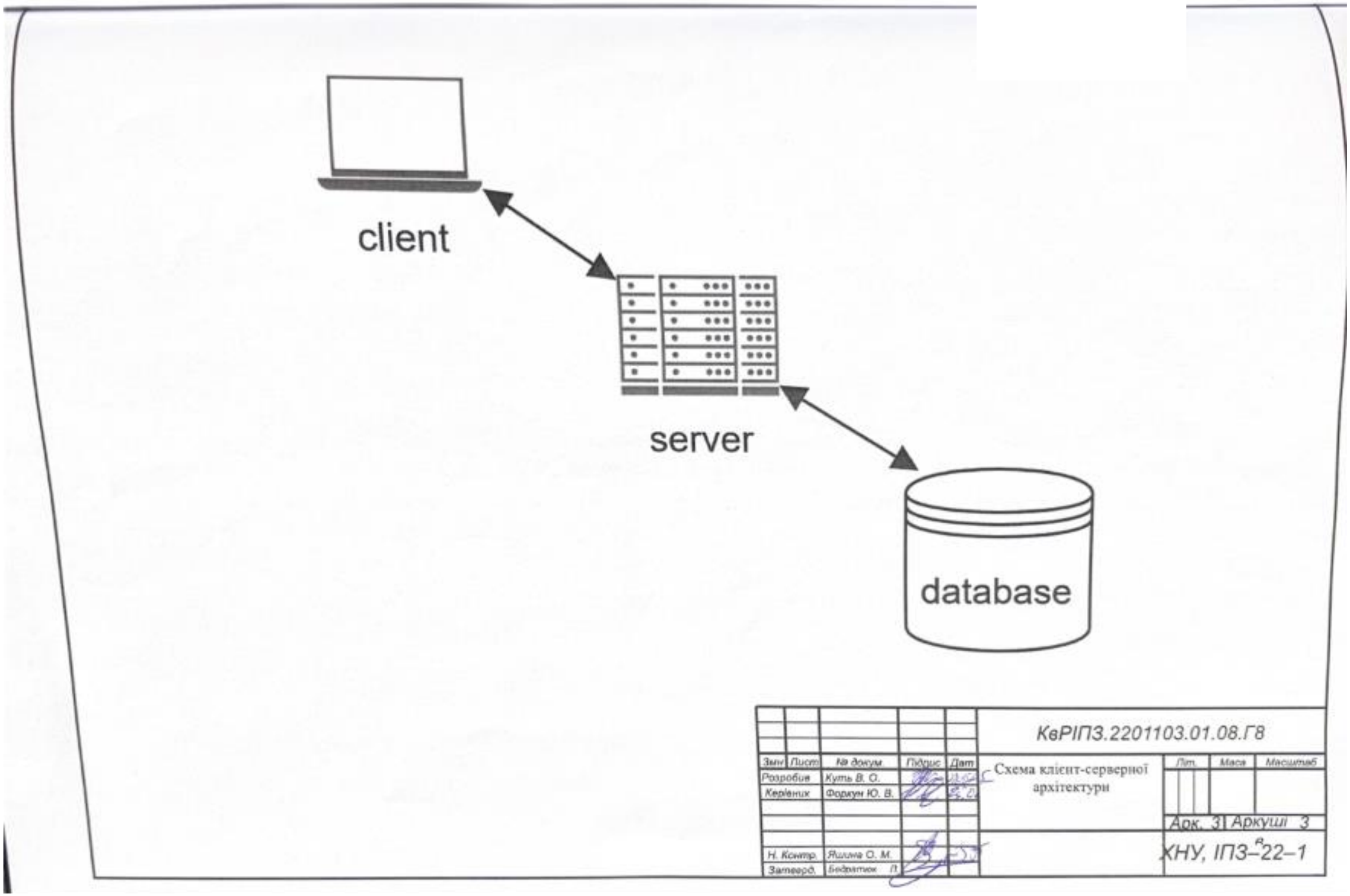
Рисунок В.14 – слайд «Дякую за увагу»

Графічні матеріали



					КвРІПЗ.2201103.01.08.Г8			
Змн.	Лист	№ докум.	Підпис	Дата				
Розробив		Кить В. О.			Схема бази даних	Літ.	Маса	Масштаб
Керівник		Форсун Ю. В.						
						Арк. 1	Аркушів 3	
Н. контр.		Янина О. М.				ХНУ, ІПЗ–22–1		
Затверд.		Бебешко Л. П.						

					КвРІПЗ.2201103.01.08.Г8		
Змн.	Лист	№ докум.	Підпис	Дат	Діаграма варіантів використання	Літ.	Маса	Масштаб
Розробив		Кутя В. О.						
Керівник		Форкун Ю. В.				Арк. 2	Аркушів 3	
Н. Контр.		Яшина О. М.				ХНУ, ІПЗ–22–1		
Затверд.		Бедратюк Л. П.						



					КвРІПЗ.2201103.01.08.Г8			
Змін	Лист	№ докум.	Підпис	Дат				
Розробив		Куть В. О.			Схема клієнт-серверної архітектури	Літ.	Маса	Масштаб
Керівник		Форкун Ю. В.						
						Арк. 31	Аркушів 3	
Н. Контр.		Яшина О. М.				ХНУ, ІПЗ-22-1		
Затверд.		Бефратюк Л.						

Супровідні документи

Завідувачу кафедри інженерії програмного
забезпечення проф. Леоніду БЕДРАТЮКУ
здобувача вищої освіти
Кутя Владислава Олександровича
факультет ІТ, IVкурс, група ІПЗ-21-1

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності в Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії Хмельницького національного університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-обчислювального комплексу StrikePlagiarism та/або програмно-технічного засобу AntiPlagiarism і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення текстових збігів у роботах.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

0306.26
дата

підпис

Anti-Plagiarism (http://ap.km.ua) v-16.718

Максимальне співпадіння з одним документом 3.0%

Словники перевірки: UA, US, RU. **Помилок в документах: 12%**

	Документ		Сумарний збіг по Базі Даних	
ID: 272699 Назва: БКР_Вебсайт для електронної комерції з продажу товарів для велосипедистів Додано в БД: 2026-05-28 Автора: Владислав КУТЬ Керівники: канд. техн. наук, доцент Форкун Юрій Вікторович Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	99564	753	4525 (5%)	67 (9%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Владислав КУТЬ

Співавтор:

Назва: Вебсайт для електронної комерції з продажу товарів для велосипедистів

Науковий керівник: канд. техн. наук, доцент Юрій ФОРКУН

Підрозділ: Кафедра інженерії програмного забезпечення

Коефіцієнт подібності 1:5.66%

Коефіцієнт подібності 2:0.59%

Мікропробіли: 13

Заміна букв: 0

Інтервали: 3

Білі знаки: 478

Дата створення звіту: 2026-05-28 22:21:21.0

Після аналізу Звіту подібності констатую наступне:

☑ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

Дата 25.05.2026 експерт Форкун Ю.В.

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

Дипломник Куть Владислав Олександрович

Тема Вебсайт для електронної комерції з продажу товарів для велосипедситів

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3; кількість сторінок записки 59
1.Короткий зміст пояснювальної записки та прийнятих рішень У кваліфікаційній роботі досліджено і проаналізовано предметну область електронної комерції у сфері велоіндустрії, визначено усі функціональні та нефункциональні вимоги до вебплатформи. Був проведений аналіз існуючих аналогів на ринку, розглянуто їх переваги та недоліки, що дозволило обґрунтувати актуальність створення нового спеціалізованого вебмагазину. Розглянуто сучасні інструменти для реалізації спроєктованих рішень, в результаті чого розроблено вебдодаток "VELO-HUB". Також було проведено тестування програми, за результатами якого доведено, що розроблене програмне забезпечення працює коректно та готове до експлуатації.

2. Висновок про відповідність роботи поставленому завданню Кваліфікаційна робота виконана відповідно до поставленого завдання та з дотриманням всіх вимог.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі обґрунтовано актуальність теми, визначено мету, об'єкт, предмет та ключові завдання дипломного проєктування. У першому розділі проведено детальний аналіз предметної області, розглянуто існуючі рішення в ніші електронної комерції та чітко визначені вимоги до майбутнього вебмагазину. У другому розділі проведено аналіз сучасних архітектурних шаблонів, обрано модель клієнт-серверної взаємодії, а також спроєктовано структуру бази даних (SQL Server) та логіку взаємодії компонентів системи. У третьому розділі підготовлено середовище розробки, виконано практичну реалізацію програмних модулів засобами інструментарію ASP.NET MVC та мови програмування C#, описано роботу модулів каталогу товарів, кошика покупця й адміністративної панелі, а також проведено тестування системи, яке підтвердило стабільність вебдодатка та його відповідність функціональним вимогам..

4. Позитивні сторони роботи Актуальність роботи зумовлена високим попитом на спеціалізовані рішення в e-commerce. До позитивних сторін варто віднести вдале застосування фреймворку ASP.NET MVC для створення масштабованої архітектури проєкту, гнучку структуру бази даних для керування характеристиками велотоварів, а також високу швидкість відгуку інтерфейсу

5. Негативні сторони роботи У поточній версії онлайн-магазину фільтрація велотоварів реалізована за основними параметрами (категорія, ціна) — було б доцільно розширити її для детальних технічних характеристик (розмір рами, діаметр коліс тощо). Також у майбутньому варто реалізувати автоматичне підключення API платіжних систем для онлайн-оплати безпосередньо на сайті

6. Оцінка графічного оформлення та пояснювальної Графічне оформлення виконано відповідно до теми роботи та наочно подано у вигляді діаграм (UML/ERD) та копій екранів інтерфейсу програми. Пояснювальна записка оформлена згідно з вимогами ДСТУ.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота заслуговує на позитивну оцінку. Матеріал пояснювальної записки структурований, логічно послідовний та чіткий. Графічний матеріал повною мірою відображає деталі реалізації та архітектуру вебмагазину.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана у повному обсязі, відповідає поставленій задачі та заслуговує на оцінку «добре».

РЕЦЕНЗЕНТ Савенко Ніна Станіславівна, д.т.н., проф., проф. каф. ІКІС ХНУ

"25." травня 2026 р. _____
(підпис)

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

першого освітнього рівня «Бакалавр»

Студента: Кутя Владислава
Група: ІПЗ-22-1

Тема: *«Вебсайт для електронної комерції з продажу товарів для велисипедстів»*

Спеціальність: 121 – Інженерія програмного забезпечення

Короткий зміст пояснювальної записки

Кваліфікаційна робота присвячена розробці веб-платформи для автоматизації торгівлі велотоварами, що відповідає сучасним вимогам ринку. Актуальність теми обумовлена зростанням попиту на цифрові рішення в індустрії велотоварів. Метою роботи є створення функціональної інформаційної системи, яка оптимізує шлях користувача та забезпечує надійну обробку транзакцій. Для досягнення цієї мети визначено завдання, серед яких аналіз предметної області, формулювання вимог до системи, проектування архітектури бази даних та програмна реалізація.

Відповідність отриманих результатів роботи поставленим завданням

Завдання, сформульовані у вступі, включають проведення аналізу предметної області формулювання вимог, проектування архітектури, програмну реалізацію та тестування. Висновки роботи підтверджують, що результати в цілому відповідають поставленим завданням, оскільки розроблена система охоплює всі ключові аспекти, зазначені в завданнях. Оцінка відповідності: **в цілому відповідають**.

Оцінка розділів

Розділ 1: Змістовий аналіз предметної області, її структурних та функціональних особливостей

Опис предметної області є логічним і структурованим, з акцентом на важливі аспекти, що свідчить про глибоке розуміння теми. Однак, відсутність прикладів конкретних товарів або сценаріїв їх використання знижує зрозумілість. Проблеми, які має вирішити програмне забезпечення, окреслені, але не сформульовані чітко, що ускладнює їх сприйняття. Структура та функції системи описані логічно, але не містять достатньої деталізації. Визначено основні категорії користувачів та їх потреби, проте не вистачає конкретних прикладів сценаріїв взаємодії. Опис вхідних/вихідних даних та безпеки охоплює важливі аспекти, але потребує детальнішого аналізу.

Розділ 2: Вибір архітектури та шаблонів проєктування

Вибір архітектури MVC обґрунтований, з акцентом на переваги, такі як гнучкість та швидкість обробки запитів. Однак, не розглянуто альтернативні архітектури, що обмежує повноту аналізу. Використання шаблону MVC відповідає задачам, але відсутні конкретні приклади застосування шаблонів проектування. Поділ на рівні описано чітко, але не вистачає графічних ілюстрацій. Основні модулі системи описані, але логіка взаємодії між ними не розкрита.

Розділ 3: Програмна реалізація модулів та контролерів системи

Описано реалізацію бази даних та логіку роботи з товарами, але не надано конкретних прикладів роботи з бізнес-процесами. Реалізація автентифікації та авторизації описана, але не вистачає прикладів токенів або middleware. Інтеграція з платіжними системами не згадана, що є суттєвим недоліком. Описано архітектуру на базі MVC, але відсутні скріншоти або фрагменти коду для візуалізації реалізації інтерфейсу.

Позитивні сторони

Кваліфікаційна робота демонструє оригінальність у підході до розробки веб-платформи для велотоварів, що відповідає сучасним вимогам ринку. Описані рішення є функціональними та логічно структурованими, що свідчить про глибоке розуміння предметної області. Використання сучасних технологій, таких як ASP.NET Core та Entity Framework, підкреслює якість реалізації.

Недоліки

Робота має суттєві недоліки, зокрема відсутність конкретних прикладів у описі предметної області, що ускладнює розуміння. Проблеми не сформульовані чітко, а деякі аспекти, такі як інтеграція з платіжними системами, не описані. Відсутність графічних ілюстрацій та прикладів коду знижує якість сприйняття матеріалу.

Відгук в цілому

Тема роботи є актуальною та має практичну значущість, оскільки відповідає потребам ринку електронної комерції. Зміст роботи в цілому відповідає темі та завданню, хоча є недоліки в деталізації. Новизна ідей та рішень присутня, але потребує більшої аргументації. Об'єктивність та обґрунтованість висновків можуть бути покращені шляхом детальнішого аналізу.

Оцінка кваліфікаційної роботи

Кваліфікаційна робота заслуговує оцінки **добре**. Вона виконана в повному обсязі з дотриманням основних вимог, але має істотні недоліки в деталізації та аргументації.

Рекомендації

Рекомендується доопрацювати роботу, зокрема уточнити формулювання проблем, додати приклади конкретних товарів та сценаріїв використання, а також розглянути інтеграцію з платіжними системами. Включення графічних ілюстрацій та прикладів коду підвищить якість роботи.

OpenAI API-асистент
Session ID: fd28663c-5799-484e-901a-cca48c1a7125
Підписано автоматично, модель gpt-4o-mini
Дата: 26.05.2026

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатами звіту/звітів перевірки роботи, продукованими програмно-технічним засобом (ами), на наявність текстових збігів.
Назва кваліфікаційної роботи: «Вебсайт для електронної комерції з продажу товарів для велосипедистів»
Автор: Куть Владислав Олександрович
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Спеціальність: 121 – Інженерія програмного забезпечення
Науковий керівник: Форкун Юрій Вікторович, кандидат технічних наук, доцент
Після аналізу звіту/звітів зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається до захисту.	**відповідає**
2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована.	
3	Виявлені запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Виявлені запозичення частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнуті. Робота може бути допущена до захисту після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) у тексті кваліфікаційної роботи системою перевірки на плагіат виявлено схожість з окремими документами переважно в частині загальновживаних обов'язкових словосполучень у стандартних бланках, структурі змісту, назвах розділів і підрозділів, рамках форм, а також у назвах та URL-адресах публікацій переліку джерел посилання;

2) запозичення, виявлені у тексті роботи, мають фрагментарний характер і не впливають на самостійність виконання кваліфікаційної роботи.

Максимальний обсяг запозичень, визначений системою Anti-Plagiarism, складає 2.0% з одного джерела. Загальна сумарна подібність у базі даних складає 5% за символами та 12% за лексемами.

Крім того, за результатами додаткового аналізу коефіцієнт подібності 1 становить 5.66%, коефіцієнт подібності 2 – 0.59%. Заміни букв, маніпуляцій з інтервалами та зайвих білих знаків не виявлено. Виявлені мікропробіли у кількості 13 мають технічний характер і не свідчать про навмисне спотворення тексту. З урахуванням наведених обґрунтувань, результати перевірки відповідають характеру теми і свідчать на користь самостійності виконання кваліфікаційної роботи.

Дата 3.06.26

Завідувач кафедри _____ Леонід БЕДРАТЮК

Гарант освітньої програми _____ Леонід БЕДРАТЮК

Керівник кваліфікаційної роботи _____ Юрій ФОРКУН